aiogram Documentation

Release 3.19.0

aiogram Team

CONTENTS

1	Features							
	1.1	Simple	usage					
	1.2	Usage v	vithout dispatcher					
2	Conte	ents	7					
	2.1	Installat	tion					
		2.1.1	From PyPI					
		2.1.2	From Arch Linux Repository					
		2.1.3	From GitHub					
	2.2	Migrati	on FAQ (2.x -> 3.0)					
		2.2.1	Dependencies					
		2.2.2	Dispatcher					
		2.2.3	Filtering events					
		2.2.4	Bot API					
		2.2.5	Middlewares					
		2.2.6	Keyboard Markup					
		2.2.7	Callbacks data					
		2.2.8	Finite State machine					
		2.2.9	Sending Files					
		2.2.10	Webhook					
		2.2.11	Telegram API Server					
		2.2.12	Telegram objects transformation (to dict, to json, from json)					
		2.2.13	ChatMember tools					
	2.3	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?					
		2.3.8	Global defaults					
	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	Finite State Machine					
		2.4.7	Middlewares					

	2.4.8	Errors
	2.4.9	Flags
	2.4.10	Webhook
o =	2.4.11	Class based handlers
2.5		635
	2.5.1	Keyboard builder
	2.5.2	Translation
	2.5.3	Chat action sender
	2.5.4	WebApp
	2.5.5	Callback answer
	2.5.6	Formatting
	2.5.7	Media group builder
	2.5.8	Deep Linking
	2.5.9	Telegram object serialization
2.6		log
	2.6.1	3.19.0 (2025-03-19)
	2.6.2	3.18.0 (2025-02-16)
	2.6.3	3.17.0 (2025-01-02)
	2.6.4	3.16.0 (2024-12-21)
	2.6.5	3.15.0 (2024-11-17)
	2.6.6	3.14.0 (2024-11-02)
	2.6.7	3.13.1 (2024-09-18)
	2.6.8	3.13.0 (2024-09-08)
	2.6.9	3.12.0 (2024-08-16)
	2.6.10	3.11.0 (2024-08-09)
	2.6.11	3.10.0 (2024-07-07)
	2.6.12	3.9.0 (2024-07-06)
	2.6.13	3.8.0 (2024-06-19)
	2.6.14	3.7.0 (2024-05-31)
	2.6.15	3.6.0 (2024-05-06)
	2.6.16	3.5.0 (2024-04-23)
	2.6.17	3.4.1 (2024-02-17)
	2.6.18	3.4.0 (2024-02-16)
	2.6.19	3.3.0 (2023-12-31)
	2.6.20	3.2.0 (2023-11-24)
	2.6.21	3.1.1 (2023-09-25)
	2.6.22	3.1.0 (2023-09-22)
		3.0.0 (2023-09-01)
	2.6.24	3.0.0rc2 (2023-08-18)
	2.6.25	3.0.0rc1 (2023-08-06)
	2.6.26	3.0.0b9 (2023-07-30)
	2.6.27	3.0.0b8 (2023-07-17)
	2.6.28	3.0.0b7 (2023-02-18)
	2.6.29	3.0.0b6 (2022-11-18)
	2.6.30	3.0.0b5 (2022-10-02)
	2.6.31	3.0.0b4 (2022-08-14)
	2.6.32	3.0.0b3 (2022-04-19)
	2.6.33	3.0.0b2 (2022-02-19)
	2.6.34	3.0.0b1 (2021-12-12)
	2.6.35	3.0.0a18 (2021-12-12)
	2.6.36	3.0.0a18 (2021-11-10)
	2.6.37	3.0.0a17 (2021-09-24)
	2.6.38	3.0.0a16 (2021-09-22)
	2.6.38	
	4.0.39	3.0.0a14 (2021-08-17)

2.6.40	2.14.3 (2021-07-21)	5
2.6.41	2.14.2 (2021-07-26)	5
2.6.42	2.14 (2021-07-27)	5
2.6.43	2.13 (2021-04-28)	5
2.6.44	2.12.1 (2021-03-22)	5
2.6.45	2.12 (2021-03-14)	7
2.6.46	2.11.2 (2021-11-10)	
2.6.47	2.11.1 (2021-11-10)	
2.6.48	2.11 (2021-11-08)	
2.6.49	2.10.1 (2021-09-14)	
2.6.50	2.10 (2021-09-13)	
2.6.51	2.9.2 (2021-06-13)	
2.6.52	2.9 (2021-06-08)	
2.6.53	2.8 (2021-04-26)	
2.6.54	2.7 (2021-04-07)	
2.6.55	2.6.1 (2021-01-25)	
2.6.56	2.6 (2021-01-23)	
2.6.57	2.5.3 (2021-01-05)	
2.6.58	2.5.2 (2021-01-01)	
2.6.59	2.5.1 (2021-01-01)	
2.6.60	2.5 (2021-01-01)	
2.6.61	2.4 (2021-10-29)	
2.6.62	2.3 (2021-08-16)	
2.6.63	2.2 (2021-06-09)	
2.6.64	2.1 (2021-04-18)	
2.6.65	2.0.1 (2021-12-31)	
2.6.66	2.0 (2021-10-28)	
2.6.67	1.4 (2021-08-03)	
2.6.68	1.3.3 (2021-07-16)	
2.6.69	1.3.2 (2021-05-27)	
2.6.70	1.3.1 (2018-05-27)	3
2.6.71	1.3 (2021-04-22)	3
2.6.72	1.2.3 (2018-04-14)	4
2.6.73	1.2.2 (2018-04-08)	4
2.6.74	1.2.1 (2018-03-25)	4
2.6.75	1.2 (2018-02-23)	4
2.6.76	1.1 (2018-01-27)	4
2.6.77	1.0.4 (2018-01-10)	5
2.6.78	1.0.3 (2018-01-07)	5
2.6.79	1.0.2 (2017-11-29)	5
2.6.80	1.0.1 (2017-11-21)	
2.6.81	1.0 (2017-11-19)	
2.6.82	0.4.1 (2017-08-03)	
2.6.83	0.4 (2017-08-05)	
2.6.84	0.3.4 (2017-08-04)	
2.6.85	0.3.3 (2017-07-05)	
2.6.86	0.3.2 (2017-07-04)	
2.6.87	0.3.1 (2017-07-04)	
2.6.88		
2.6.89	0.1 (2017-06-03)	
	uting	
2.7.1	Developing	
2.7.2	Star on GitHub	
2.7.3	Guides	J

2.7

	Take answers Funding								
Python Module Index							711		
Index									717

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 8.3 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)

A Warning

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

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

- @aiogram
- @aiogramua
- @aiogram_uz
- @aiogram_kz
- @aiogram_ru
- @aiogram_fa
- @aiogram_it
- @aiogram_br

1.1 Simple usage

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

(continues on next page)

(continued from previous page)

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

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

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

1.2 Usage without dispatcher

Just only interact with Bot API, without handling events

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

(continues on next page)

(continued from previous page)

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

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

6 Chapter 1. Features

CHAPTER

TWO

CONTENTS

2.1 Installation

2.1.1 From PyPI

pip install -U aiogram

2.1.2 From Arch Linux Repository



Warning

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

pacman -S python-aiogram

Development build

2.1.3 From GitHub

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

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



This guide is still in progress.

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

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

1 Note

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

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

2.2.1 Dependencies

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

pip install aiogram[i18n]

2.2.2 Dispatcher

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

2.2.3 Filtering events

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

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

2.2.4 Bot API

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

2.2.5 Middlewares

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

2.2.6 Keyboard Markup

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

2.2.7 Callbacks data

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

2.2.8 Finite State machine

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

• The state proxy is deprecated; you should update the state data by calling state.set_data(...) and state. get_data() respectively.

2.2.9 Sending Files

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

2.2.10 Webhook

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

2.2.11 Telegram API Server

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

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

- Methods TelegramObject.to_object(), TelegramObject.to_json() and TelegramObject.to_python() have been removed due to the use of pydantic models.
- TelegramObject.to_object() should be replaced by TelegramObject.model_validate() (Read more)
- TelegramObject.as_json() should be replaced by aiogram.utils.serialization. deserialize_telegram_object_to_python()
- <TelegramObject>.to_python() should be replaced by json.dumps(deserialize_telegram_object_to_python(<Tele

Here are some usage examples:

Creating an object from a dictionary representation of an object

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

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

• Creating a json representation of an object

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

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

• Creating a dictionary representation of an object

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

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

2.2.13 ChatMember tools

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

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

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

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

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

```
# Version 2.x

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

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

```
# Version 3.x

from aiogram.utils.chat_member import ADMINS, MEMBERS

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

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

1 Note

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

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

2.3 Bot API

aiogram now is fully support of Telegram Bot API

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

2.3.1 Bot

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

This class has aliases for all methods and named in lower_camel_case.

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

A Warning

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

Bases: object

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

Bot class

Parameters

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

Raises

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

property token: str

property id: int

Get bot ID from token

Returns

context($auto_close: bool = True$) \rightarrow AsyncIterator[Bot]

Generate bot context

Parameters

auto_close - close session on exit

Returns

async me() $\rightarrow User$

Cached alias for getMe method

Returns

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

Download file by file path to destination.

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

Parameters

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

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

Download file by file id or Downloadable object to destination.

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

Parameters

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

2.3.2 Client session

Client sessions is used for interacting with API server.

Use Custom API server

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

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

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

    method – Method instance

                    • timeout – Request timeout
```

2.3. Bot API 15

Returns

```
Raises
```

TelegramApiError -

Stream reader

aiohttp

AiohttpSession represents a wrapper-class around ClientSession from aiohttp

Currently AiohttpSession is a default session used in aiogram.Bot

```
class aiogram.client.session.aiohttp.AiohttpSession(proxy: Iterable[str | Tuple[str, BasicAuth]] | str |
Tuple[str, BasicAuth] | None = None, limit: int = 100, **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)
```

1 Note

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

Authorization

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

Consider examples:

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

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

or simply include your basic auth credential in URL

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



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

Proxy chains

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

Example of chain proxies:

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

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

Client session middlewares

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

Some useful cases that is:

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

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

How to register client session middleware?

Register using register method

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

Register using decorator

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

Example

Class based session middleware

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



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

Function based session middleware

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

2.3.3 Types

Here is list of all available API types:

Available types

Animation

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

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

```
file id: str
```

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

```
file_unique_id: str
```

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

width: int

Video width as defined by the sender

height: int

Video height as defined by the sender

duration: int

Duration of the video in seconds as defined by the sender

thumbnail: PhotoSize | None

Optional. Animation thumbnail as defined by the sender

file_name: str | None

Optional. Original animation filename as defined by the sender

mime_type: str | None

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

file_size: int | None

Optional. File size in bytes. It can be bigger than 2^31 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

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

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

file_id: str

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

file_unique_id: str

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

duration: int

Duration of the audio in seconds as defined by the sender

performer: str | None

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

title: str | None

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

file_name: str | None

Optional. Original filename as defined by the sender

mime_type: str | None

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

file_size: int | None

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

thumbnail: *PhotoSize* | None

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

BackgroundFill

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

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

- aiogram.types.background_fill_solid.BackgroundFillSolid
- aiogram.types.background_fill_gradient.BackgroundFillGradient
- $\bullet \ aiogram.types.background_fill_freeform_gradient.BackgroundFillFreeformGradient\\$

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

BackgroundFillFreeformGradient

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

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

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

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

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

Type of the background fill, always 'freeform_gradient'

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

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'

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

BackgroundTypeChatTheme

class aiogram.types.background_type_chat_theme.BackgroundTypeChatTheme(*, type:

Literal['chat_theme'] =
'chat_theme',
theme_name: str,
**extra data: Any)

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'

theme_name: str

Name of the chat theme, which is usually an emoji

BackgroundTypeFill

class aiogram.types.background_type_fill.BackgroundTypeFill(*, type: Literal['fill'] = 'fill', fill:

BackgroundFillSolid |
BackgroundFillGradient |
BackgroundFillFreeformGradient,
dark_theme_dimming: int,
**extra_data: Any)

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: BackgroundFillUnion

The background fill

dark_theme_dimming: int

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

BackgroundTypePattern

class aiogram.types.background_type_pattern.BackgroundTypePattern(*, type: Literal['pattern'] =

'pattern', document:
Document, fill:
BackgroundFillSolid |
BackgroundFillGradient |
BackgroundFillFreeformGradient, intensity: int,
is_inverted: bool | None =
None, is_moving: bool | None
= None, **extra_data: Any)

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

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

type: Literal['pattern']

Type of the background, always 'pattern'

document: Document

Document with the pattern

fill: BackgroundFillUnion

The background fill that is combined with the pattern

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

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

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.

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
- $\bullet \ aiogram.types.bot_command_scope_all_group_chats.BotCommandScopeAllGroupChats\\$
- 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

BotCommandScopeAllChatAdministrators

class aiogram.types.bot_command_scope_all_chat_administrators.BotCommandScopeAllChatAdministrators(*,

eral[I = Bot-Command

> Scope **extra_d

type: Lit-

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*

BotCommandScopeAllGroupChats

```
 \textbf{class} \  \, \text{aiogram.types.bot\_command\_scope\_all\_group\_chats.} \\ \textbf{BotCommandScopeAllGroupChats}(*, type: \\ Lit-\\ eral[BotCommandScopeTy \\ = Bot-\\ Com-\\ mand-\\ ScopeType.ALL\_GROUP\_\\ **ex-\\ tra\_data: \\ Any)
```

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

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

type: Literal[BotCommandScopeType.ALL_GROUP_CHATS]

Scope type, must be all_group_chats

BotCommandScopeAllPrivateChats

```
class aiogram.types.bot_command_scope_all_private_chats.BotCommandScopeAllPrivateChats(*,
```

```
type:
Lit-
eral[BotCommandSc
=
Bot-
Com-
mand-
ScopeType.ALL_PRF
**ex-
```

tra_data: Any)

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

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

type: Literal[BotCommandScopeType.ALL_PRIVATE_CHATS]

Scope type, must be all_private_chats

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

chat_id: ChatIdUnion

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

chat_id: ChatIdUnion

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

BotCommandScopeChatMember

```
\textbf{class} \ \texttt{aiogram.types.bot\_command\_scope\_chat\_member.} \textbf{BotCommandScopeChatMember} (*, \textit{type: Lit-posterior}) and \texttt{aiogram.types.bot\_command\_scope\_chat\_member.} (*, \textit{type: Lit-posterior}) and \texttt{ai
```

```
r(*, type: Lit-
eral[BotCommandScopeType.CHAT
= BotCommand-
ScopeType.CHAT_MEMBER,
chat_id: int | str,
user_id: int,
**extra_data:
Any)
```

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: ChatIdUnion

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

BotCommandScopeDefault

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

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

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

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

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.

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

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

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.

message_ids: list[int]

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

BusinessOpeningHours

class aiogram.types.business_opening_hours.BusinessOpeningHours(*, time_zone_name: str,

opening_hours:
list[BusinessOpeningHoursInterval],

**extra_data: Any)

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

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

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 aiogram.methods.answer_callback_query.AnswerCallbackQuery. It is, therefore, necessary to react by calling aiogram.methods.answer_callback_query.AnswerCallbackQuery even if no notification to the user is needed (e.g., without specifying any of the optional parameters).

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

id: str

Unique identifier for this query

from_user: User

Sender

chat_instance: str

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

message: MaybeInaccessibleMessageUnion | None

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

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) <math>\rightarrow AnswerCallbackOuery
```

Shortcut for method aiogram.methods.answer_callback_query.AnswerCallbackQuery 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 | ReactionTypePaid] | None = None,
                                     background custom emoji id: str | None = None, bio: str | None = None,
                                     birthdate: Birthdate | None = None, business intro: BusinessIntro | None =
                                     None, business location: BusinessLocation | None = None,
                                     business opening hours: BusinessOpeningHours | None = None,
                                     can_set_sticker_set: bool | None = None, custom_emoji_sticker_set_name:
                                     str \mid None = None, description: str \mid None = None,
                                     emoji_status_custom_emoji_id: str | None = None,
                                     emoji status expiration date: datetime serializer, return type=int,
                                     when\_used=unless - none) | None = None,
                                     has\_aggressive\_anti\_spam\_enabled: bool \mid None = None,
                                     has_hidden_members: bool | None = None, has_private_forwards: bool |
                                     None = None, has\_protected\_content: bool | None = None,
                                     has restricted voice and video messages: bool | None = None,
                                     has_visible_history: bool | None = None, invite_link: str | None = None,
                                     join by request: bool | None = None, join to send messages: bool | None =
                                     None, linked_chat_id: int | None = None, location: ChatLocation | None =
                                     None, message auto delete time: int | None = None, permissions:
                                     ChatPermissions | None = None, personal_chat: Chat | None = None, photo:
                                     ChatPhoto | None = None, pinned message: Message | None = None,
                                     profile accent color id: int | None = None,
                                     profile background custom emoji id: str | None = None, slow mode delay:
                                     int \mid None = None, sticker\_set\_name: str \mid None = None,
                                     unrestrict_boost_count: int | None = None, **extra_data: Any)
```

This object represents a chat.

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

id: int

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

type: str

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

title: str | None

Optional. Title, for supergroups, channels and group chats

username: str | None

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

first_name: str | None

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

last_name: str | None

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

is_forum: bool | None

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

accent_color_id: int | None

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

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

active_usernames: list[str] | None

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

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

available_reactions: list[ReactionTypeUnion] | 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 <code>aiogram.methods.get_chat.GetChat</code>.

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

pinned_message: Message | None

Optional. The most recent pinned message (by sending date). Returned only in aiogram.methods. get_chat.GetChat.

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

profile_accent_color_id: int | None

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

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

profile_background_custom_emoji_id: str | None

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

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

slow_mode_delay: int | None

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

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

sticker_set_name: str | None

Optional. For supergroups, name of group sticker set. Returned only in aiogram.methods.get_chat. GetChat.

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

unrestrict_boost_count: int | None

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

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

property shifted_id: int

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

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

property full_name: str

Get full name of the Chat.

For private chat it is first_name + last_name. For other chat types it is title.

$ban_sender_chat(sender_chat_id: int, **kwargs: Any) \rightarrow BanChatSenderChat$

 $Shortcut\ for\ method\ a \emph{iogram.methods.ban_chat_sender_chat.} Ban \textit{ChatSenderChat}\ will\ automatically\ fill\ method\ attributes:$

• 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) $\rightarrow UnbanChatSenderChat$

Shortcut for method <code>aiogram.methods.unban_chat_sender_chat.UnbanChatSenderChat</code> will automatically fill method attributes:

• chat_id

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

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

Parameters

sender_chat_id – Unique identifier of the target sender chat

Returns

instance of method aiogram.methods.unban_chat_sender_chat.
UnbanChatSenderChat

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

Shortcut for method $aiogram.methods.get_chat_administrators.GetChatAdministrators$ will automatically fill method attributes:

• chat_id

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

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

Returns

instance of method aiogram.methods.get_chat_administrators.
GetChatAdministrators

$delete_message(message_id: int, **kwargs: Any) \rightarrow DeleteMessage$

Shortcut for method $aiogram.methods.delete_message.DeleteMessage$ will automatically fill method attributes:

• chat_id

Use this method to delete a message, including service messages, with the following limitations:

- A message can only be deleted if it was sent less than 48 hours ago.
- Service messages about a supergroup, channel, or forum topic creation can't be deleted.
- A dice message in a private chat can only be deleted if it was sent more than 24 hours ago.
- Bots can delete outgoing messages in private chats, groups, and supergroups.
- Bots can delete incoming messages in private chats.
- Bots granted can_post_messages permissions can delete outgoing messages in channels.
- If the bot is an administrator of a group, it can delete any message there.
- If the bot has can_delete_messages permission in a supergroup or a channel, it can delete any message there.

Returns True on success.

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

Parameters

message_id - Identifier of the message to delete

instance of method aiogram.methods.delete_message.DeleteMessage

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

Shortcut for method <code>aiogram.methods.revoke_chat_invite_link.RevokeChatInviteLink</code> 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

instance of method aiogram.methods.revoke_chat_invite_link.
RevokeChatInviteLink

```
edit_invite_link(invite_link: str, name: str | None = None, expire_date: DateTimeUnion | None = None, member_limit: int | None = None, creates_join_request: bool | None = None, **kwargs: Any) \rightarrow EditChatInviteLink
```

Shortcut for method <code>aiogram.methods.edit_chat_invite_link.EditChatInviteLink</code> will automatically fill method attributes:

chat_id

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

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

Parameters

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

Returns

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

```
\begin{tabular}{ll} \textbf{create_invite_link} (name: str \mid None = None, expire_date: DateTimeUnion \mid None = None, \\ member_limit: int \mid None = None, creates_join_request: bool \mid None = None, \\ **kwargs: Any) \rightarrow CreateChatInviteLink \\ \end{tabular}
```

 $Shortcut\ for\ method\ aiogram.methods.create_chat_invite_link.CreateChatInviteLink\ will\ automatically\ fill\ method\ attributes:$

• 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 <code>aiogram.methods.export_chat_invite_link.ExportChatInviteLink</code> will automatically fill method attributes:

• chat_id

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

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

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

Returns

instance of method aiogram.methods.export_chat_invite_link.
ExportChatInviteLink

do(action: str, business_connection_id: str | None = None, message_thread_id: int | None = None, **kwargs: Any) \rightarrow SendChatAction

Shortcut for method <code>aiogram.methods.send_chat_action.SendChatAction</code> will automatically fill method attributes:

• chat id

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

Example: The ImageBot needs some time to process a request and upload the image. Instead of sending a text message along the lines of 'Retrieving image, please wait...', the bot may use <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

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 <code>aiogram.methods.delete_chat_sticker_set.DeleteChatStickerSet</code> 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 $can_set_sticker_set$ optionally returned in $aiogram.methods.get_chat.GetChat$ requests to check if the bot can use this method. Returns True on success.

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

Parameters

sticker_set_name – Name of the sticker set to be set as the group sticker set

Returns

instance of method aiogram.methods.set_chat_sticker_set.SetChatStickerSet

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

Shortcut for method $aiogram.methods.get_chat_member.GetChatMember$ will automatically fill method attributes:

• chat_id

Use this method to get information about a member of a chat. The method is only guaranteed to work for other users if the bot is an administrator in the chat. Returns a <code>aiogram.types.chat_member.ChatMember</code> 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

Returns

 $\begin{tabular}{ll} in stance & of & method & aiogram.methods.unpin_all_chat_messages. \\ \textit{UnpinAllChatMessages} & \end{tabular}$

 $unpin_message(business_connection_id: str \mid None = None, message_id: int \mid None = None, **kwargs: Any) \rightarrow UnpinChatMessage$

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

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

Returns

instance of method aiogram.methods.unpin_chat_message.UnpinChatMessage

 $pin_message(message_id: int, business_connection_id: str | None = None, disable_notification: bool | None = None, **kwargs: Any) <math>\rightarrow PinChatMessage$

Shortcut for method <code>aiogram.methods.pin_chat_message.PinChatMessage</code> will automatically fill method attributes:

• chat id

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

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

Parameters

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

Returns

instance of method aiogram.methods.pin_chat_message.PinChatMessage

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

Shortcut for method aiogram.methods.set_chat_administrator_custom_title. SetChatAdministratorCustomTitle will automatically fill method attributes:

• chat_id

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

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

Parameters

• user_id - Unique identifier of the target user

 custom_title – New custom title for the administrator; 0-16 characters, emoji are not allowed

Returns

instance of method aiogram.methods.set_chat_administrator_custom_title.
SetChatAdministratorCustomTitle

set_permissions(permissions: ChatPermissions, use_independent_chat_permissions: bool | None = None, **kwargs: Any) \rightarrow SetChatPermissions

Shortcut for method 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: DateTimeUnion | None = None, **kwargs: Any) → 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

Returns

instance of method aiogram.methods.restrict_chat_member.RestrictChatMember

 $unban(user_id: int, only_if_banned: bool \mid None = None, **kwargs: Any) \rightarrow UnbanChatMember$

Shortcut for method aiogram.methods.unban_chat_member.UnbanChatMember will automatically fill method attributes:

• chat_id

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

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

Parameters

- user_id Unique identifier of the target user
- only_if_banned Do nothing if the user is not banned

Returns

instance of method aiogram.methods.unban_chat_member.UnbanChatMember

ban(user_id: int, until_date: DateTimeUnion | None = None, revoke_messages: bool | None = None, **kwargs: Any) → BanChatMember

Shortcut for method $aiogram.methods.ban_chat_member.BanChatMember$ will automatically fill method attributes:

• chat_id

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

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

Parameters

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

Returns

 $instance\ of\ method\ aiogram.methods.ban_chat_member.BanChatMember$

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

Shortcut for method <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 <code>aiogram.methods.set_chat_title.SetChatTitle</code> 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.methods.delete_chat_photo.DeleteChatPhoto

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

Shortcut for method <code>aiogram.methods.set_chat_photo.SetChatPhoto</code> will automatically fill method attributes:

• chat_id

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

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

Parameters

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

Returns

instance of method aiogram.methods.set_chat_photo.SetChatPhoto

 $\label{local_continuity} \begin{tabular}{ll} unpin_all_general_forum_topic_messages (**kwargs: Any) \rightarrow UnpinAllGeneralForumTopicMessages \\ Shortcut for method aiogram.methods.unpin_all_general_forum_topic_messages. \\ UnpinAllGeneralForumTopicMessages will automatically fill method attributes: \\ \end{tabular}$

• 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: $bool \mid None = None,$ **extra_data: Any)

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

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

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

 $\textbf{class} \ a iogram. types. chat_background. \textbf{ChatBackground}(*, \textit{type:}\ BackgroundTypeFill\ |\ Albert ChatBackground(*, \textit{type:}\ ChatBackgroundTypeFill\ |\ Albert ChatBackground(*, \textit{type:}\ ChatBackgrou$

BackgroundTypeWallpaper | BackgroundTypePattern |

BackgroundTypeChatTheme, **extra_data: Any)

This object represents a chat background.

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

type: BackgroundTypeUnion

Type of the background

ChatBoost

class aiogram.types.chat_boost.ChatBoost(*, boost_id: str, add_date: _datetime_serializer,

return_type=int, when_used=unless - none)], expiration_date: datetime serializer, return type=int, when used=unless -

none)], source: ChatBoostSourcePremium |

ChatBoostSourceGiftCode | ChatBoostSourceGiveaway,

**extra_data: Any)

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

expiration_date: DateTime

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

source: ChatBoostSourceUnion

Source of the added boost

ChatBoostAdded

class aiogram.types.chat_boost_added.ChatBoostAdded(*, boost_count: int, **extra_data: Any)

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

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

boost_count: int

Number of boosts added by the user

ChatBoostRemoved

This object represents a boost removed from a chat.

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

chat: Chat

Chat which was boosted

boost id: str

Unique identifier of the boost

remove_date: DateTime

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

source: ChatBoostSourceUnion

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
- $\bullet \ aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGiftCode\\$
- aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGiveaway

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

ChatBoostSourceGiftCode

= ChatBoostSource-Type.GIFT_CODE, 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'

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,
```

Iype.GIVEAWAY,
giveaway_message_id:
int, user: User | None =
None, prize_star_count:
int | None = None,
is_unclaimed: bool |
None = None,
**extra_data: Any)

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

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

source: Literal[ChatBoostSourceType.GIVEAWAY]

Source of the boost, always 'giveaway'

giveaway_message_id: int

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

```
user: User | None
```

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

```
prize_star_count: int | None
```

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

is_unclaimed: bool | None

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

ChatBoostSourcePremium

```
class aiogram.types.chat_boost_source_premium.ChatBoostSourcePremium(*, source: Lit-
```

eral[ChatBoostSourceType.PREMIUM]
= ChatBoostSource-

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'

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

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 | ReactionTypePaid/ | None = None, background custom emoji id: str | None = None, bio: $str \mid None = None$, birthdate: Birthdate | *None* = *None*, *business_intro*: BusinessIntro | None = None, business location: BusinessLocation | None = None, business_opening_hours: BusinessOpeningHours | None = None, can_set_sticker_set: bool | None = None, custom_emoji_sticker_set_name: str | None = *None*, description: str | None = None, emoji_status_custom_emoji_id: str | None = None, emoji status expiration date: datetime serializer, return_type=int, when_used=unless - none)] | None = None, has aggressive anti spam enabled: bool None = None, has_hidden_members: bool | None = None, has private forwards: $bool \mid None = None$, $has_protected_content: bool \mid None = None,$ 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 | $None = None, linked_chat_id: int | None = None,$ location: ChatLocation | None = None, $message_auto_delete_time: int | None = None,$ permissions: ChatPermissions | None = None, $personal_chat$: Chat | None = None, photo: ChatPhoto | *None* = *None*, *pinned_message*: Message | None = None, profile_accent_color_id: int | None = None, profile background custom emoji id: str | $None = None, slow_mode_delay: int | None = None,$ sticker set name: $str \mid None = None$, unrestrict_boost_count: int | None = None, max reaction count: int, can send gift: bool | None = None, can_send_paid_media: bool | None = None, **extra data: Any)

This object contains full information about a chat.

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

id: int

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

type: str

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

accent_color_id: int

Identifier of the accent color for the chat name and backgrounds of the chat photo, reply header, and link

preview. See accent colors for more details. max_reaction_count: int The maximum number of reactions that can be set on a message in the chat title: str | None Optional. Title, for supergroups, channels and group chats username: str | None Optional. Username, for private chats, supergroups and channels if available first_name: str | None Optional. First name of the other party in a private chat last_name: str | None Optional. Last name of the other party in a private chat is_forum: bool | None Optional. True, if the supergroup chat is a forum (has topics enabled) photo: ChatPhoto | None Optional. Chat photo active_usernames: list[str] | None Optional. If non-empty, the list of all active chat usernames; for private chats, supergroups and channels birthdate: Birthdate | None Optional. For private chats, the date of birth of the user business_intro: BusinessIntro | None Optional. For private chats with business accounts, the intro of the business business_location: BusinessLocation | None Optional. For private chats with business accounts, the location of the business business_opening_hours: BusinessOpeningHours | None Optional. For private chats with business accounts, the opening hours of the business personal_chat: Chat | None Optional. For private chats, the personal channel of the user available_reactions: list[ReactionTypeUnion] | 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

bio: str | None

Optional. Bio of the other party in a private chat

has_private_forwards: bool | None

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

has_restricted_voice_and_video_messages: bool | None

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

join_to_send_messages: bool | None

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

join_by_request: bool | None

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

description: str | None

Optional. Description, for groups, supergroups and channel chats

invite_link: str | None

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

pinned_message: Message | None

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

permissions: ChatPermissions | None

Optional. Default chat member permissions, for groups and supergroups

can_send_gift: bool | None

Optional. True, if gifts can be sent to the chat

can_send_paid_media: bool | None

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

slow_mode_delay: int | None

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

unrestrict_boost_count: int | None

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

message_auto_delete_time: int | None

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

has_aggressive_anti_spam_enabled: bool | None

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

has_hidden_members: bool | None

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

has_protected_content: bool | None

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

has_visible_history: bool | None

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

sticker_set_name: str | None

Optional. For supergroups, name of the group sticker set

can_set_sticker_set: bool | None

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

custom_emoji_sticker_set_name: str | None

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

linked_chat_id: int | None

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

location: ChatLocation | None

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

ChatInviteLink

class aiogram.types.chat_invite_link.ChatInviteLink(*, invite_link: str, creator: User,

creates_join_request: bool, is_primary: bool, is_revoked: bool, name: str | None = None, expire_date: _datetime_serializer, return_type=int, when_used=unless - none)] | None = None, member_limit: int | None = None, pending_join_request_count: int | None = None, subscription_period: int | None = None, subscription_price: int | None = None, **extra data: Any)

Represents an invite link for a chat.

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

invite_link: str

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

creator: *User*

Creator of the link

creates_join_request: bool

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

is_primary: bool

True, if the link is primary

is_revoked: bool

True, if the link is revoked

name: str | None

Optional. Invite link name

expire_date: DateTime | None

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

member_limit: int | None

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

pending_join_request_count: int | None

Optional. Number of pending join requests created using this link

subscription_period: int | None

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

subscription_price: int | None

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

ChatJoinRequest

Represents a join request sent to a chat.

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

chat: Chat

Chat to which the request was sent

from_user: User

User that sent the join request

user_chat_id: int

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

date: DateTime

Date the request was sent in Unix time

bio: str | None

Optional. Bio of the user.

invite_link: ChatInviteLink | None

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

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

Shortcut for method <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

Returns

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

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

Shortcut for method $aiogram.methods.decline_chat_join_request.DeclineChatJoinRequest$ will automatically fill method attributes:

- chat_id
- user_id

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

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

Returns

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

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

Shortcut for method $aiogram.methods.send_message.SendMessage$ will automatically fill method attributes:

• chat_id

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

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

Parameters

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- parse_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse_mode*
- link_preview_options Link preview generation options for the message
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_message.SendMessage

```
answer_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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id:
Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_message.SendMessage

Shortcut for method aiogram.methods.send_animation.SendAnimation will automatically fill method attributes:

• chat id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent aiogram.types.message.Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- animation Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*»
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_animation.SendAnimation

Shortcut for method aiogram.methods.send_animation.SendAnimation will automatically fill method attributes:

• chat_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent *aiogram.types.message*. Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- animation Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data. More information on Sending Files »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **duration** Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.

- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_animation.SendAnimation

answer_audio(audio: InputFileUnion, business_connection_id: Optional[str] = None, 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, 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')>,

allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,

reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] =

None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendAudio

Shortcut for method <code>aiogram.methods.send_audio.SendAudio</code> will automatically fill method attributes:

• chat_id

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

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

Parameters

• audio – Audio file to send. Pass a file_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »

- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_audio.SendAudio

Shortcut for method <code>aiogram.methods.send_audio.SendAudio</code> will automatically fill method attributes:

· chat id

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

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

Parameters

- audio Audio file to send. Pass a file_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_audio.SendAudio

answer_contact(phone_number: str, first_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, last_name: Optional[str] = None, vcard: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendContact

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

• chat_id

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

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

Parameters

- phone_number Contact's phone number
- **first_name** Contact's first name
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_contact.SendContact

answer_contact_pm(phone_number: str, first_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, last_name: Optional[str] = None, vcard: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendContact

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

• chat id

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

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

- **phone_number** Contact's phone number
- first_name Contact's first name
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to

- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_contact.SendContact

Shortcut for method <code>aiogram.methods.send_document.SendDocument</code> will automatically fill method attributes:

chat_id

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

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

Parameters

- document File to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More information on Sending Files »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*»
- **caption** Document caption (may also be used when resending documents by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the document caption. See formatting options for more details.

- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **disable_content_type_detection** Disables automatic server-side content type detection for files uploaded using multipart/form-data
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_document.SendDocument

Shortcut for method <code>aiogram.methods.send_document.SendDocument</code> will automatically fill method attributes:

• chat id

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_document.SendDocument

 $\begin{tabular}{ll} \textbf{answer_game} (game_short_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[InlineKeyboardMarkup] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) \rightarrow SendGame \\ \end{tabular}$

Shortcut for method aiogram.methods.send_game.SendGame will automatically fill method attributes:

• chat_id

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

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

Parameters

- game_short_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_game.SendGame

```
 \begin{array}{ll} \textbf{answer\_game\_pm}(game\_short\_name: str, business\_connection\_id: Optional[str] = None, \\ message\_thread\_id: Optional[int] = None, disable\_notification: Optional[bool] = None, \\ protect\_content: Optional[Union[bool, Default]] = & Optional['protect\_content') >, \\ allow\_paid\_broadcast: Optional[bool] = None, message\_effect\_id: Optional[str] = None, \\ reply\_parameters: Optional[ReplyParameters] = None, reply\_markup: \\ Optional[InlineKeyboardMarkup] = None, allow\_sending\_without\_reply: Optional[bool] \\ = None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow SendGame \\ \end{array}
```

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_game.SendGame

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

• chat_id

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

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

Parameters

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- currency Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

- prices Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- provider_token Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider_data JSON-serialized data about the invoice, which will be shared with the
 payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo_size** Photo size in bytes
- photo_width Photo width
- photo_height Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- **need_email** Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need_shipping_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- is_flexible Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable_notification Sends the message silently. Users will receive a notification with no sound.

74

- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_invoice.SendInvoice

```
answer_invoice_pm(title: str, description: str, payload: str, currency: str, prices: list[LabeledPrice],
                      message thread id: Optional[int] = None, provider token: Optional[str] = None,
                      max_tip_amount: Optional[int] = None, suggested_tip_amounts: Optional[list[int]] =
                      None, start parameter: Optional[str] = None, provider data: Optional[str] = None,
                      photo_url: Optional[str] = None, photo_size: Optional[int] = None, photo_width:
                      Optional[int] = None, photo height: Optional[int] = None, need name:
                      Optional[bool] = None, need\_phone\_number: Optional[bool] = None, need\_email:
                      Optional[bool] = None, need shipping address: Optional[bool] = None,
                      send_phone_number_to_provider: Optional[bool] = None, send_email_to_provider:
                      Optional[bool] = None, is\_flexible: Optional[bool] = None, disable\_notification:
                      Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =
                       <Default('protect content')>, allow paid broadcast: Optional[bool] = None,
                      message\_effect\_id: Optional[str] = None, reply\_parameters:
                      Optional[ReplyParameters] = None, reply_markup: Optional[InlineKeyboardMarkup]
                      = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id:
                      Optional[int] = None, **kwargs: Any) \rightarrow SendInvoice
```

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

• chat_id

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

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

Parameters

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider_data JSON-serialized data about the invoice, which will be shared with the
 payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo_size** Photo size in bytes
- **photo_width** Photo width
- photo_height Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- need_shipping_address Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_invoice.SendInvoice

Shortcut for method aiogram.methods.send_location.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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_location.SendLocation

Shortcut for method <code>aiogram.methods.send_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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_location.SendLocation

```
 \begin{tabular}{ll} answer_media_group (media: list[MediaUnion], 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')>, allow\_paid\_broadcast: Optional[bool] = None, \\ message\_effect\_id: Optional[str] = None, reply\_parameters: \\ Optional[ReplyParameters] = None, allow\_sending\_without\_reply: Optional[bool] = \\ None, reply\_to\_message\_id: Optional[int] = None, **kwargs: Any) \rightarrow \\ SendMediaGroup \\ \end{tabular}
```

Shortcut for method <code>aiogram.methods.send_media_group.SendMediaGroup</code> will automatically fill method attributes:

• chat_id

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

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

Parameters

• media – A JSON-serialized array describing messages to be sent, must include 2-10 items

- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable_notification** Sends messages silently. Users will receive a notification with no sound.
- **protect_content** Protects the contents of the sent messages from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the messages are a reply, ID of the original message

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

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

- 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

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the messages are a reply, ID of the original message

instance of method aiogram.methods.send_media_group.SendMediaGroup

answer_photo(photo: InputFileUnion, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendPhoto

Shortcut for method <code>aiogram.methods.send_photo.SendPhoto</code> will automatically fill method attributes:

• chat_id

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

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

Parameters

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- **business_connection_id** Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media

- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_photo.SendPhoto

Shortcut for method <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

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_photo.SendPhoto

answer_poll(question: str, options: list[InputPollOptionUnion], 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[DateTimeUnion] = None,
is_closed: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None,
reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendPoll

Shortcut for method aiogram.methods.send_poll.SendPoll will automatically fill method attributes:

• chat_id

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

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

- question Poll question, 1-300 characters
- options A JSON-serialized list of 2-10 answer options
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **question_parse_mode** Mode for parsing entities in the question. See formatting options for more details. Currently, only custom emoji entities are allowed
- **question_entities** A JSON-serialized list of special entities that appear in the poll question. It can be specified instead of *question_parse_mode*
- is_anonymous True, if the poll needs to be anonymous, defaults to True
- type Poll type, 'quiz' or 'regular', defaults to 'regular'
- allows_multiple_answers True, if the poll allows multiple answers, ignored for polls in quiz mode, defaults to False
- correct_option_id 0-based identifier of the correct answer option, required for polls in quiz mode
- **explanation** Text that is shown when a user chooses an incorrect answer or taps on the lamp icon in a quiz-style poll, 0-200 characters with at most 2 line feeds after entities parsing
- **explanation_parse_mode** Mode for parsing entities in the explanation. See formatting options for more details.
- **explanation_entities** A JSON-serialized list of special entities that appear in the poll explanation. It can be specified instead of *explanation_parse_mode*
- **open_period** Amount of time in seconds the poll will be active after creation, 5-600. Can't be used together with *close_date*.
- **close_date** Point in time (Unix timestamp) when the poll will be automatically closed. Must be at least 5 and no more than 600 seconds in the future. Can't be used together with *open_period*.
- **is_closed** Pass True if the poll needs to be immediately closed. This can be useful for poll preview.
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_poll.SendPoll

answer_poll_pm(question: str, options: list[InputPollOptionUnion], 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[DateTimeUnion] = None, is_closed: Optional[bool] =
None, disable_notification: Optional[bool] = None, protect_content:
Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast:
Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters:
Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] =
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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_poll.SendPoll

Shortcut for method 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_dice.SendDice

```
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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_dice.SendDice

Shortcut for method aiogram.methods.send_sticker.SendSticker will automatically fill method attributes:

• chat_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

- **sticker** Sticker to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/formdata. *More information on Sending Files* ». Video and animated stickers can't be sent via an HTTP URL.
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- emoji Emoji associated with the sticker; only for just uploaded stickers
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_sticker.SendSticker

Shortcut for method aiogram.methods.send_sticker.SendSticker will automatically fill method attributes:

• chat_id

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_sticker.SendSticker

answer_venue(latitude: float, longitude: float, title: str, address: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, foursquare_id: Optional[str] = None, foursquare_type: Optional[str] = None, google_place_id: Optional[str] = None, google_place_type: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVenue

Shortcut for method <code>aiogram.methods.send_venue.SendVenue</code> will automatically fill method attributes:

• chat id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

- 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

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_venue.SendVenue

```
answer_venue_pm(latitude: float, longitude: float, title: str, address: str, business_connection_id:

Optional[str] = None, message_thread_id: Optional[int] = None, foursquare_id:

Optional[str] = None, foursquare_type: Optional[str] = None, google_place_id:

Optional[str] = None, google_place_type: Optional[str] = None, disable_notification:

Optional[bool] = None, protect_content: Optional[Union[bool, Default]] =

<Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None,

message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters]

= None, reply_markup: Optional[ReplyMarkupUnion] = None,

allow_sending_without_reply: Optional[bool] = None, reply_to_message_id:

Optional[int] = None, **kwargs: Any) → SendVenue
```

Shortcut for method <code>aiogram.methods.send_venue.SendVenue</code> will automatically fill method attributes:

• chat_id

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

Parameters

- latitude Latitude of the venue
- longitude Longitude of the venue
- title Name of the venue
- address Address of the venue
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **foursquare_id** Foursquare identifier of the venue
- **foursquare_type** Foursquare type of the venue, if known. (For example, 'arts_entertainment/default', 'arts_entertainment/aquarium' or 'food/icecream'.)
- google_place_id Google Places identifier of the venue

- google_place_type Google Places type of the venue. (See supported types.)
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_venue.SendVenue

answer_video(video: InputFileUnion, 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, cover:

Optional[InputFileUnion] = None, start_timestamp: Optional[DateTimeUnion] = None,
caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = 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

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*
- **cover** Cover for the video in the message. 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* »
- **start_timestamp** Start timestamp for the video in the message
- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports_streaming** Pass True if the uploaded video is suitable for streaming
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply_to_message_id - If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_video.SendVideo

Shortcut for method <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

- 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*
- **cover** Cover for the video in the message. 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 »*

- **start_timestamp** Start timestamp for the video in the message
- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports_streaming** Pass True if the uploaded video is suitable for streaming
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_video.SendVideo

Shortcut for method <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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_video_note.SendVideoNote

Shortcut for method <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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_video_note.SendVideoNote

answer_voice(voice: InputFileUnion, business_connection_id: Optional[str] = None, 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, duration: Optional[int] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = < Default('protect_content') >, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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

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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_voice.SendVoice

Shortcut for method aiogram.methods.send_voice.SendVoice will automatically fill method attributes:

• chat_id

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message.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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_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.

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

ChatMemberAdministrator

class aiogram.types.chat_member_administrator.ChatMemberAdministrator(*, status: Lit-

eral[ChatMemberStatus.ADMINISTRATOR = ChatMemberSta-

tus.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:

booi, can_invite_users.

bool, can_post_stories:

bool, can_edit_stories:

bool, can_delete_stories:

bool,

can_post_messages:

 $bool \mid None = None,$

can_edit_messages: bool

| None = None,

can_pin_messages: bool

| None = None, can_manage_topics:

bool | None = None, custom_title: str | None

= None, **extra_data:

Any)

Represents a chat member that has some additional privileges.

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

status: Literal[ChatMemberStatus.ADMINISTRATOR]

The member's status in the chat, always 'administrator'

user: User

Information about the user

can_be_edited: bool

True, if the bot is allowed to edit administrator privileges of that user

is_anonymous: bool

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

can_manage_chat: bool

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

can_delete_messages: bool

True, if the administrator can delete messages of other users

can_manage_video_chats: bool

True, if the administrator can manage video chats

can_restrict_members: bool

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

can_promote_members: bool

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

can_change_info: bool

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

can_invite_users: bool

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

can_post_stories: bool

True, if the administrator can post stories to the chat

can_edit_stories: bool

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

can_delete_stories: bool

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

can_post_messages: bool | None

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

can_edit_messages: bool | None

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

can_pin_messages: bool | None

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

can_manage_topics: bool | None

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

custom_title: str | None

Optional. Custom title for this user

ChatMemberBanned

class aiogram.types.chat_member_banned.ChatMemberBanned(*, status:

Literal[ChatMemberStatus.KICKED] = ChatMemberStatus.KICKED, user: User, until_date: _datetime_serializer, return_type=int, when_used=unless - none)], **extra_data: Any)

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

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

status: Literal[ChatMemberStatus.KICKED]

The member's status in the chat, always 'kicked'

user: *User*

Information about the user

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'

user: User

Information about the user

ChatMemberMember

class aiogram.types.chat_member_member.ChatMemberMember(*, status:

Literal[ChatMemberStatus.MEMBER] = ChatMemberStatus.MEMBER, user: User, until_date: _datetime_serializer, return_type=int, when_used=unless - none)] | None = None, **extra_data: Any)

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

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

status: Literal[ChatMemberStatus.MEMBER]

The member's status in the chat, always 'member'

user: *User*

Information about the user

until_date: DateTime | None

Optional. Date when the user's subscription will expire; Unix time

ChatMemberOwner

class aiogram.types.chat_member_owner.ChatMemberOwner(*, status:

Literal[ChatMemberStatus.CREATOR] =
ChatMemberStatus.CREATOR, user: User,
is_anonymous: bool, custom_title: str | None
= None, **extra_data: Any)

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

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

status: Literal[ChatMemberStatus.CREATOR]

The member's status in the chat, always 'creator'

user: User

Information about the user

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_serializer, return_type=int, when_used=unless - none)], **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

can_send_voice_notes: bool

True, if the user is allowed to send voice notes

can_send_polls: bool

True, if the user is allowed to send polls

can_send_other_messages: bool

True, if the user is allowed to send animations, games, stickers and use inline bots

can_add_web_page_previews: bool

True, if the user is allowed to add web page previews to their messages

can_change_info: bool

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

can_invite_users: bool

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

can_pin_messages: bool

True, if the user is allowed to pin messages

can_manage_topics: bool

True, if the user is allowed to create forum topics

until_date: DateTime

Date when restrictions will be lifted for this user; Unix time. If 0, then the user is restricted forever

ChatMemberUpdated

class aiogram.types.chat_member_updated.ChatMemberUpdated(*, chat: Chat, from_user: User, date:

_datetime_serializer, return_type=int,

when_used=unless - none)],

old_chat_member: ChatMemberOwner |

ChatMemberAdministrator |

ChatMember |

ChatMemberRestricted |

ChatMemberLeft | ChatMemberBanned,

new_chat_member: ChatMemberOwner

| ChatMemberAdministrator |

ChatMember |

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: ChatMemberUnion

Previous information about the chat member

new_chat_member: ChatMemberUnion

New information about the chat member

invite link: ChatInviteLink | None

Optional. Chat invite link, which was used by the user to join the chat; for joining by invite link events only.

via_join_request: bool | None

Optional. True, if the user joined the chat after sending a direct join request without using an invite link and being approved by an administrator

via_chat_folder_invite_link: bool | None

Optional. True, if the user joined the chat via a chat folder invite link

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

Optional[list[MessageEntity]] = None, link_preview_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link_preview')>, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_message.SendMessage

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

- animation Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data. More information on Sending Files »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.

- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_animation.SendAnimation

answer_audio(audio: InputFileUnion, business_connection_id: Optional[str] = None, 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, 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')>,

allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,

reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[ReplyMarkupUnion] = 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

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_audio.SendAudio

answer_contact (phone_number: str, first_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, last_name: Optional[str] = None, vcard: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendContact

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

• chat_id

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

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

Parameters

- phone_number Contact's phone number
- first_name Contact's first name
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_contact.SendContact

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_document.SendDocument

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')>, allow_paid_broadcast:

Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup: Optional[InlineKeyboardMarkup] =

None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id:

Optional[int] = None, **kwargs: Any) → SendGame

Shortcut for method aiogram.methods.send_game.SendGame will automatically fill method attributes:

chat id

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

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

Parameters

- game_short_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_game.SendGame

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

• chat_id

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

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

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max tip amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent

- message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider_data JSON-serialized data about the invoice, which will be shared with the
 payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo_size** Photo size in bytes
- photo_width Photo width
- **photo_height** Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need_shipping_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_invoice.SendInvoice

Shortcut for method <code>aiogram.methods.send_location</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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_location.SendLocation

Shortcut for method <code>aiogram.methods.send_media_group.SendMediaGroup</code> will automatically fill method attributes:

• chat_id

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

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

Parameters

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable_notification Sends messages silently. Users will receive a notification with no sound.
- **protect_content** Protects the contents of the sent messages from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the messages are a reply, ID of the original message

Returns

instance of method aiogram.methods.send_media_group.SendMediaGroup

answer_photo(photo: InputFileUnion, business_connection_id: Optional[str] = None, message_thread_id:

Optional[int] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str,

Default]] = <Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] =

None, show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,

disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool,

Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None,

message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] =

None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply:

Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) →

SendPhoto

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

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to

- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_photo.SendPhoto

Shortcut for method aiogram.methods.send_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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_poll.SendPoll

Shortcut for method <code>aiogram.methods.send_dice.SendDice</code> will automatically fill method attributes:

• chat id

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

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

Parameters

120

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_dice.SendDice

Shortcut for method 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_sticker.SendSticker

answer_venue(latitude: float, longitude: float, title: str, address: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, foursquare_id: Optional[str] = None, foursquare_type: Optional[str] = None, google_place_id: Optional[str] = None, google_place_type: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_venue.SendVenue

```
answer_video(video: InputFileUnion, 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, cover:

Optional[InputFileUnion] = None, start_timestamp: Optional[DateTimeUnion] = None,
caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] =
None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVideo
```

Shortcut for method aiogram.methods.send_video.SendVideo will automatically fill method attributes:

• chat_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message

is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

- video Video to send. Pass a file_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. More information on Sending Files »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- width Video width
- height Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*"""
- **cover** Cover for the video in the message. 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* »
- **start_timestamp** Start timestamp for the video in the message
- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- supports_streaming Pass True if the uploaded video is suitable for streaming
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_video.SendVideo

Shortcut for method $aiogram.methods.send_video_note.SendVideoNote$ will automatically fill method attributes:

• chat id

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- **video_note** Video note to send. Pass a file_id as String to send a video note that exists on the Telegram servers (recommended) or upload a new video using multipart/form-data. *More information on Sending Files* ». Sending video notes by a URL is currently unsupported
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*

2.3. Bot API 125

>>>

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_video_note.SendVideoNote

answer_voice(voice: InputFileUnion, business_connection_id: Optional[str] = None, 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, duration: Optional[int] = None, disable_notification: Optional[bool] = None,

protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,

allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,

reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[ReplyMarkupUnion] = 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

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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_voice.SendVoice

ChatPermissions

class aiogram.types.chat_permissions.**ChatPermissions**(*, can_send_messages: bool | None = None,

can_send_audios: bool | None = None,
can_send_documents: bool | None = None,
can_send_photos: bool | None = None,
can_send_videos: bool | None = None,
can_send_video_notes: bool | None = None,
can_send_voice_notes: bool | None = None,
can_send_polls: bool | None = None,
can_send_other_messages: bool | None =
None, can_add_web_page_previews: bool |
None = None, can_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

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.

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.

```
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

```
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

CopyTextButton

```
class aiogram.types.copy_text_button.CopyTextButton(*, text: str, **extra_data: Any)
```

This object represents an inline keyboard button that copies specified text to the clipboard.

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

text: str

The text to be copied to the clipboard; 1-256 characters

Dice

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

class aiogram.types.dice.DiceEmoji

```
DICE = ''

DART = ''

BASKETBALL = ''

FOOTBALL = ''

SLOT_MACHINE = ''

BOWLING = ''
```

Document

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

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

file_id: str

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

file_unique_id: str

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

thumbnail: PhotoSize | None

Optional. Document thumbnail as defined by the sender

file_name: str | None

Optional. Original filename as defined by the sender

mime_type: str | None

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

file_size: int | None

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

ExternalReplyInfo

class aiogram.types.external_reply_info.ExternalReplyInfo(*, origin: MessageOriginUser |

MessageOriginHiddenUser | MessageOriginChat | MessageOriginChannel, chat: Chat | None = None, message_id: int | None = *None*, *link_preview_options*: $LinkPreviewOptions \mid None = None,$ animation: Animation | None = None, *audio:* Audio | None = None, document: Document | *None* = *None*, *paid_media*: PaidMediaInfo | *None* = *None*, *photo*: *list*[PhotoSize] | *None* = *None*, *sticker*: Sticker | None = None, story: Story | None = None, video: Video | None =None, video note: VideoNote | None = None, voice: Voice | None = None, has_media_spoiler: bool | 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: MessageOriginUnion
     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
     Optional. Message is an audio file, information about the file
document: Document | None
     Optional. Message is a general file, information about the file
paid_media: PaidMediaInfo | None
     Optional. Message contains paid media; information about the paid media
photo: list[PhotoSize] | None
     Optional. Message is a photo, available sizes of the photo
sticker: Sticker | None
     Optional. Message is a sticker, information about the sticker
story: Story | None
     Optional. Message is a forwarded story
video: Video | None
     Optional. Message is a video, information about the video
video_note: VideoNote | None
     Optional. Message is a video note, information about the video message
voice: Voice | None
     Optional. Message is a voice message, information about the file
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.

```
file_size: int | None
```

Optional. File size in bytes. It can be bigger than 2³¹ and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this value.

```
file_path: str | None
```

Optional. File path. Use https://api.telegram.org/file/bot<token>/<file_path> to get the file.

ForceReply

Upon receiving a message with this object, Telegram clients will display a reply interface to the user (act as if the user has selected the bot's message and tapped 'Reply'). This can be extremely useful if you want to create user-friendly step-by-step interfaces without having to sacrifice privacy mode. Not supported in channels and for messages sent on behalf of a Telegram Business account.

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

- Explain the user how to send a command with parameters (e.g. /newpoll question answer1 answer2). May be appealing for hardcore users but lacks modern day polish.
- Guide the user through a step-by-step process. 'Please send me your question', 'Cool, now let's add the first answer option', 'Great. Keep adding answer options, then send /done when you're ready'.

The last option is definitely more attractive. And if you use aiogram.types.force_reply. 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

```
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

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

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

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

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

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

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

Giveaway

This object represents a message about a scheduled giveaway.

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

chats: list[Chat]

The list of chats which the user must join to participate in the giveaway

winners_selection_date: DateTime

Point in time (Unix timestamp) when winners of the giveaway will be selected

winner_count: int

The number of users which are supposed to be selected as winners of the giveaway

only_new_members: bool | None

Optional. True, if only users who join the chats after the giveaway started should be eligible to win

has_public_winners: bool | None

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

prize_description: str | None

Optional. Description of additional giveaway prize

country_codes: list[str] | None

Optional. A list of two-letter ISO 3166-1 alpha-2 country codes indicating the countries from which eligible users for the giveaway must come. If empty, then all users can participate in the giveaway. Users with a phone number that was bought on Fragment can always participate in giveaways.

prize_star_count: int | None

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

premium_subscription_month_count: int | None

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

GiveawayCompleted

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

```
giveaway_message: Message | None
```

Optional. Message with the giveaway that was completed, if it wasn't deleted

```
is_star_giveaway: bool | None
```

Optional. True, if the giveaway is a Telegram Star giveaway. Otherwise, currently, the giveaway is a Telegram Premium giveaway.

GiveawayCreated

This object represents a service message about the creation of a scheduled giveaway.

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

```
prize_star_count: int | None
```

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

GiveawayWinners

```
class aiogram.types.giveaway_winners.GiveawayWinners(*, chat: Chat, giveaway_message_id: int, winners_selection_date: _datetime_serializer, return_type=int, when_used=unless - none)], winner_count: int, winners: list[User], additional_chat_count: int | None = None, prize_star_count: int | None = None, premium_subscription_month_count: int | None = None, unclaimed_prize_count: int | None = None, only_new_members: bool | None = None, was_refunded: bool | None = None, prize_description: str | 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

prize_star_count: int | None

Optional. The number of Telegram Stars that were split between giveaway winners; for Telegram Star giveaways only

premium_subscription_month_count: int | None

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

unclaimed_prize_count: int | None

Optional. Number of undistributed prizes

only_new_members: bool | None

Optional. True, if only users who had joined the chats after the giveaway started were eligible to win

was_refunded: bool | None

Optional. True, if the giveaway was canceled because the payment for it was refunded

prize_description: str | None

Optional. Description of additional giveaway prize

InaccessibleMessage

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

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

chat: Chat

Chat the message belonged to

message_id: int

Unique message identifier inside the chat

date: Literal[0]

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

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

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

Shortcut for method $aiogram.methods.send_message.SendMessage$ will automatically fill method attributes:

• chat_id

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

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

Parameters

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- parse_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse_mode*
- link_preview_options Link preview generation options for the message
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_message.SendMessage

```
reply(text: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities: Optional[list[MessageEntity]] = None, link_preview_options: Optional[Union[LinkPreviewOptions, Default]] = <Default('link_preview')>, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, disable_web_page_preview: Optional[Union[bool, Default]] = <Default('link_preview_is_disabled')>, **kwargs: Any) → SendMessage
```

Shortcut for method $aiogram.methods.send_message.SendMessage$ will automatically fill method attributes:

- chat id
- reply_parameters

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

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

- text Text of the message to be sent, 1-4096 characters after entities parsing
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- parse_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse_mode*
- link_preview_options Link preview generation options for the message
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message

instance of method aiogram.methods.send_message.SendMessage

Shortcut for method aiogram.methods.send_animation.SendAnimation will automatically fill method attributes:

• chat_id

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent aiogram.types.message.Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- animation Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data.

 *More information on Sending Files >> *
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.

- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_animation.SendAnimation

Shortcut for method aiogram.methods.send_animation.SendAnimation will automatically fill method attributes:

- chat_id
- reply_parameters

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent <code>aiogram.types.message</code> is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

• **animation** – Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to

get an animation from the Internet, or upload a new animation using multipart/form-data. *More information on Sending Files »*

- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*""
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_animation.SendAnimation

answer_audio(audio: InputFileUnion, business_connection_id: Optional[str] = None, 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, 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')>,

allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,

reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[ReplyMarkupUnion] = 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

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

- 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

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_audio.SendAudio

reply_audio(audio: InputFileUnion, business_connection_id: Optional[str] = None, 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, 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')>,

allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,

reply_markup: Optional[ReplyMarkupUnion] = 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
- reply_parameters

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

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

Parameters

- audio Audio file to send. Pass a file_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Audio caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the audio caption. See formatting options for more details.

- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*»
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_audio.SendAudio

 $\begin{array}{l} \textbf{answer_contact}(phone_number:\ str,\ first_name:\ str,\ business_connection_id:\ Optional[str] = None,\\ message_thread_id:\ Optional[int] = None,\ last_name:\ Optional[str] = None,\ vcard:\\ Optional[str] = None,\ disable_notification:\ Optional[bool] = None,\ protect_content:\\ Optional[Union[bool,\ Default]] = & Optional['protect_content') >,\ allow_paid_broadcast:\\ Optional[bool] = None,\ message_effect_id:\ Optional[str] = None,\ reply_parameters:\\ Optional[ReplyParameters] = None,\ reply_markup:\ Optional[ReplyMarkupUnion] =\\ None,\ allow_sending_without_reply:\ Optional[bool] = None,\ reply_to_message_id:\\ Optional[int] = None,\ **kwargs:\ Any) \rightarrow SendContact \end{aligned}$

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

• chat id

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

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

- phone_number Contact's phone number
- **first_name** Contact's first name

- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_contact.SendContact

reply_contact(phone_number: str, first_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, last_name: Optional[str] = None, vcard: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content') >, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendContact

Shortcut for method aiogram.methods.send_contact.SendContact will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

Parameters

- phone_number Contact's phone number
- **first_name** Contact's first name
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent

- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- last_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_contact.SendContact

Shortcut for method <code>aiogram.methods.send_document.SendDocument</code> will automatically fill method attributes:

• chat_id

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

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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_document.SendDocument

Shortcut for method <code>aiogram.methods.send_document.SendDocument</code> will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_document.SendDocument

 $\begin{tabular}{ll} \textbf{answer_game} & (game_short_name: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = & Optional[vprotect_content') > , allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[InlineKeyboardMarkup] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) \rightarrow SendGame \end{tabular}$

Shortcut for method aiogram.methods.send_game.SendGame will automatically fill method attributes:

· chat id

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

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

Parameters

- game_short_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_game.SendGame

 $\textbf{reply_game} (\textit{game_short_name}: \textit{str}, \textit{business_connection_id}: Optional[\textit{str}] = None, \textit{message_thread_id}: \\ Optional[int] = None, \textit{disable_notification}: Optional[bool] = None, \textit{protect_content}: \\ Optional[Union[bool, Default]] = <Default('protect_content')>, \textit{allow_paid_broadcast}: \\ Optional[bool] = None, \textit{message_effect_id}: Optional[\textit{str}] = None, \textit{reply_markup}: \\ Optional[InlineKeyboardMarkup] = None, \textit{allow_sending_without_reply}: Optional[bool] = \\ None, **kwargs: Any) \rightarrow \textit{SendGame}$

Shortcut for method aiogram.methods.send_game.SendGame will automatically fill method attributes:

chat_id

• reply_parameters

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

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

Parameters

- game_short_name Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_game.SendGame

Shortcut for method <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

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- currency Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider_data JSON-serialized data about the invoice, which will be shared with the
 payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- photo_size Photo size in bytes
- photo_width Photo width
- photo_height Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- **need_email** Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need_shipping_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_invoice.SendInvoice

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

- chat_id
- reply_parameters

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

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

Parameters

• **title** – Product name, 1-32 characters

- **description** Product description, 1-255 characters
- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- **prices** Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **provider_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- **provider_data** JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo_size** Photo size in bytes
- photo_width Photo width
- photo_height Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- **need_email** Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need_shipping_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.

- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_invoice.SendInvoice

Shortcut for method <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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_location.SendLocation

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

- chat_id
- reply_parameters

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- latitude Latitude of the location
- longitude Longitude of the location

- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- horizontal_accuracy The radius of uncertainty for the location, measured in meters;
 0-1500
- **live_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- proximity_alert_radius For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_location.SendLocation

Shortcut for method $aiogram.methods.send_media_group.SendMediaGroup$ 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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the messages are a reply, ID of the original message

instance of method aiogram.methods.send_media_group.SendMediaGroup

```
 \begin{split} \textbf{reply\_media\_group}(media: list[MediaUnion], 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') >, allow\_paid\_broadcast: Optional[bool] = None, \\ message\_effect\_id: Optional[str] = None, allow\_sending\_without\_reply: \\ Optional[bool] = None, **kwargs: Any) \rightarrow SendMediaGroup \end{split}
```

Shortcut for method $aiogram.methods.send_media_group.SendMediaGroup$ will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

Parameters

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **disable_notification** Sends messages silently. Users will receive a notification with no sound.

- protect_content Protects the contents of the sent messages from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_media_group.SendMediaGroup

answer_photo(photo: InputFileUnion, business_connection_id: Optional[str] = None, message_thread_id:

Optional[int] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str,

Default]] = <Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] =

None, show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,

disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool,

Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None,

message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] =

None, reply_markup: Optional[ReplyMarkupUnion] = 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

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

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

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- caption_entities A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media

- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_photo.SendPhoto

 $\begin{aligned} \textbf{reply_photo}(\textit{photo: InputFileUnion, business_connection_id: Optional[str]} &= None, message_thread_id: \\ Optional[int] &= None, caption: Optional[str] &= None, parse_mode: Optional[Union[str, Default]] &= < Default('parse_mode') >, caption_entities: Optional[list[MessageEntity]] &= \\ None, show_caption_above_media: Optional[Union[bool, Default]] &= \\ &< Default('show_caption_above_media') >, has_spoiler: Optional[bool] &= None, \\ & disable_notification: Optional[bool] &= None, protect_content: Optional[Union[bool, Default]] \\ &= < Default('protect_content') >, allow_paid_broadcast: Optional[bool] &= None, \\ & message_effect_id: Optional[str] &= None, reply_markup: Optional[ReplyMarkupUnion] &= \\ & None, allow_sending_without_reply: Optional[bool] &= None, **kwargs: Any) \rightarrow SendPhoto \end{aligned}$

Shortcut for method <code>aiogram.methods.send_photo.SendPhoto</code> will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

Parameters

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only

- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_photo.SendPhoto

answer_poll(question: str, options: list[InputPollOptionUnion], 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[DateTimeUnion] = None,
is_closed: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

• reply_to_message_id - If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_poll.SendPoll

Shortcut for method <code>aiogram.methods.send_poll</code>. <code>SendPoll</code> will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_poll.SendPoll

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_dice.SendDice

reply_dice(business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, emoji: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendDice

Shortcut for method aiogram.methods.send_dice.SendDice will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_dice.SendDice

Shortcut for method <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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to

- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_sticker.SendSticker

Shortcut for method <code>aiogram.methods.send_sticker.SendSticker</code> will automatically fill method attributes:

- chat_id
- reply_parameters

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

• allow_sending_without_reply - Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_sticker.SendSticker

answer_venue (latitude: float, longitude: float, title: str, address: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, foursquare_id: Optional[str] = None, foursquare_type: Optional[str] = None, google_place_id: Optional[str] = None, google_place_type: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to

- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_venue.SendVenue

reply_venue(latitude: float, longitude: float, title: str, address: str, business_connection_id: Optional[str] = None, message_thread_id: Optional[int] = None, foursquare_id: Optional[str] = None, foursquare_type: Optional[str] = None, google_place_id: Optional[str] = None, google_place_type: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVenue

Shortcut for method aiogram.methods.send_venue.SendVenue will automatically fill method attributes:

- chat_id
- reply_parameters

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_venue.SendVenue

answer_video(video: InputFileUnion, 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, cover:

Optional[InputFileUnion] = None, start_timestamp: Optional[DateTimeUnion] = None,
caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] =
None, reply to message id: Optional[int] = None, **kwargs: Any) → SendVideo

Shortcut for method aiogram.methods.send_video.SendVideo will automatically fill method attributes:

• chat_id

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* »

- **cover** Cover for the video in the message. 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* »
- **start_timestamp** Start timestamp for the video in the message
- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports_streaming** Pass True if the uploaded video is suitable for streaming
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

172

instance of method aiogram.methods.send_video.SendVideo

```
reply_video (video: InputFileUnion, 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, cover:

Optional[InputFileUnion] = None, start_timestamp: Optional[DateTimeUnion] = None,
caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode') >, caption_entities: Optional[list[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media') >, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content') >,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_markup: Optional[ReplyMarkupUnion] = 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
- reply_parameters

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*
- **cover** Cover for the video in the message. 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* »
- **start_timestamp** Start timestamp for the video in the message

- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports_streaming** Pass True if the uploaded video is suitable for streaming
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

 $instance\ of\ method\ aiogram.methods.send_video.Send Video$

```
 \begin{tabular}{ll} \textbf{answer\_video\_note} & (video\_note: InputFileUnion, business\_connection\_id: Optional[str] = None, \\ & message\_thread\_id: Optional[int] = None, duration: Optional[int] = None, length: \\ & Optional[int] = None, thumbnail: Optional[InputFile] = None, disable\_notification: \\ & Optional[bool] = None, protect\_content: Optional[Union[bool, Default]] = \\ & < Default('protect\_content') >, allow\_paid\_broadcast: Optional[bool] = None, \\ & message\_effect\_id: Optional[str] = None, reply\_parameters: \\ & Optional[ReplyParameters] = None, reply\_markup: Optional[ReplyMarkupUnion] = \\ & None, allow\_sending\_without\_reply: Optional[bool] = None, reply\_to\_message\_id: \\ & Optional[int] = None, **kwargs: Any) \rightarrow SendVideoNote \\ \end{tabular}
```

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_video_note.SendVideoNote

Shortcut for method $aiogram.methods.send_video_note.SendVideoNote$ will automatically fill method attributes:

- chat id
- reply_parameters

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- video_note Video note to send. Pass a file_id as String to send a video note that exists
 on the Telegram servers (recommended) or upload a new video using multipart/form-data.
 More information on Sending Files ». Sending video notes by a URL is currently unsupported
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_video_note.SendVideoNote

answer_voice(voice: InputFileUnion, business_connection_id: Optional[str] = None, 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, duration: Optional[int] = None, disable_notification: Optional[bool] = None,

protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,

allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,

reply_parameters: Optional[ReplyParameters] = None, reply_markup:

Optional[ReplyMarkupUnion] = 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

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

 $instance\ of\ method\ aiogram.methods.send_voice.SendVoice$

reply_voice(voice: InputFileUnion, business_connection_id: Optional[str] = None, 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, duration: Optional[int] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_markup: Optional[ReplyMarkupUnion] = 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
- reply_parameters

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- **voice** Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- message_thread_id Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- caption Voice message caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **duration** Duration of the voice message in seconds
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

• allow_sending_without_reply - Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_voice.SendVoice

```
answer_paid_media(star_count: int, media: list[InputPaidMediaUnion], business_connection_id: str |
None = None, payload: str | None = None, caption: str | None = None, parse_mode:
str | None = None, caption_entities: list[MessageEntity] | None = None,
show_caption_above_media: bool | None = None, disable_notification: bool | None =
None, protect_content: bool | None = None, allow_paid_broadcast: bool | None =
None, reply_parameters: ReplyParameters | None = None, reply_markup:
ReplyMarkupUnion | None = None, **kwargs: Any) → SendPaidMedia
```

Shortcut for method aiogram.methods.send_paid_media.SendPaidMedia will automatically fill method attributes:

• chat_id

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

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

Parameters

- **star_count** The number of Telegram Stars that must be paid to buy access to the media; 1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **payload** Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- caption Media caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

Returns

instance of method aiogram.methods.send_paid_media.SendPaidMedia

```
reply_paid_media(star_count: int, media: list[InputPaidMediaUnion], business_connection_id: str | None = None, payload: str | None = None, caption: str | None = None, parse_mode: str | None = None, caption_entities: list[MessageEntity] | None = None, show_caption_above_media: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, allow_paid_broadcast: bool | None = None, reply markup: ReplyMarkupUnion | None = None, **kwargs: Any) → SendPaidMedia
```

Shortcut for method <code>aiogram.methods.send_paid_media.SendPaidMedia</code> will automatically fill method attributes:

- chat_id
- reply_parameters

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

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

Parameters

- **star_count** The number of Telegram Stars that must be paid to buy access to the media; 1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- business_connection_id Unique identifier of the business connection on behalf of which the message will be sent
- **payload** Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **caption** Media caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

Returns

instance of method aiogram.methods.send_paid_media.SendPaidMedia

```
 \textbf{as\_reply\_parameters}(allow\_sending\_without\_reply: Optional[Union[bool, Default]] = \\ < Default('allow\_sending\_without\_reply')>, quote: Optional[str] = None, \\ quote\_parse\_mode: Optional[Union[str, Default]] = < Default('parse\_mode')>, \\ quote\_entities: Optional[List[MessageEntity]] = None, quote\_position: \\ Optional[int] = None) \rightarrow ReplyParameters
```

InlineKeyboardButton

class aiogram.types.inline_keyboard_button.InlineKeyboardButton(*, text: str, url: str | None =

None, callback_data: str | None
= None, web_app: WebAppInfo
| None = None, login_url:
LoginUrl | None = None,
switch_inline_query: str | None
= None,
switch_inline_query_current_chat:
str | None = None,
switch_inline_query_chosen_chat:
SwitchInlineQueryChosenChat |
None = None, copy_text:
CopyTextButton | None = None,
callback_game: CallbackGame |
None = None, pay: bool | None
= None, **extra_data: Any)

This object represents one button of an inline keyboard. Exactly one of the optional fields must be used to specify type of the button.

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

text: str

Label text on the button

url: str | None

Optional. HTTP or tg:// URL to be opened when the button is pressed. Links tg://user?id=<user_id>can be used to mention a user by their identifier without using a username, if this is allowed by their privacy settings.

callback_data: str | None

Optional. Data to be sent in a callback query to the bot when the button is pressed, 1-64 bytes

web_app: WebAppInfo | None

Optional. Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method <code>aiogram.methods.answer_web_app_query.AnswerWebAppQuery</code>. Available only in private chats between a user and the bot. Not supported for messages sent on behalf of a Telegram Business account.

login_url: LoginUrl | None

Optional. An HTTPS URL used to automatically authorize the user. Can be used as a replacement for the Telegram Login Widget.

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.

```
copy_text: CopyTextButton | None
```

Optional. Description of the button that copies the specified text to the clipboard.

```
callback_game: CallbackGame | None
```

Optional. Description of the game that will be launched when the user presses the button.

```
pay: bool | None
```

Optional. Specify True, to send a Pay button. Substrings 'and 'XTR' in the buttons's text will be replaced with a Telegram Star icon.

InlineKeyboardMarkup

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

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

```
inline_keyboard: list[list[InlineKeyboardButton]]
```

Array of button rows, each represented by an Array of aiogram.types.inline_keyboard_button. InlineKeyboardButton objects

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]

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

InputMediaAnimation

```
class aiogram.types.input_media_animation.InputMediaAnimation(*, type: ~typ-
                                                                                ing.Literal[InputMediaType.ANIMATION]
                                                                                = InputMediaType.ANIMATION,
                                                                                media: str |
                                                                                ~aiogram.types.input_file.InputFile,
                                                                                thumbnail:
                                                                                ~aiogram.types.input_file.InputFile
                                                                                | None = None, caption: str | None
                                                                                = None, parse\_mode: str \mid
                                                                                ~aiogram.client.default.Default |
                                                                                None = \langle Default('parse\_mode') \rangle,
                                                                                caption entities:
                                                                                list[~aiogram.types.message_entity.MessageEntity]
                                                                                | None = None,
                                                                                show_caption_above_media: bool |
                                                                                ~aiogram.client.default.Default |
                                                                                None = < De-
                                                                               fault('show_caption_above_media')>,
                                                                                width: int \mid None = None, height:
                                                                                int \mid None = None, duration: int \mid
                                                                                None = None, has\_spoiler: bool |
                                                                                None = None, **extra\_data:
```

~typing.Any)

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

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

type: Literal[InputMediaType.ANIMATION]

Type of the result, must be animation

media: InputFileUnion

File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file_attach_name>' to upload a new one using multipart/form-data under <file_attach_name> name. *More information on Sending Files* »

thumbnail: InputFile | None

Optional. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported serverside. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files* »

caption: str | None

Optional. Caption of the animation to be sent, 0-1024 characters after entities parsing

```
parse_mode: str | Default | None
```

Optional. Mode for parsing entities in the animation caption. See formatting options for more details.

```
caption_entities: list[MessageEntity] | None
```

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

```
show_caption_above_media: bool | Default | None
```

Optional. Pass True, if the caption must be shown above the message media

width: int | None

Optional. Animation width

height: int | None

Optional. Animation height

duration: int | None

Optional. Animation duration in seconds

has_spoiler: bool | None

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

InputMediaAudio

```
class aiogram.types.input_media_audio.InputMediaAudio(*, type:
```

```
~typing.Literal[InputMediaType.AUDIO] =
InputMediaType.AUDIO, media: str |
~aiogram.types.input_file.InputFile,
thumbnail:
~aiogram.types.input_file.InputFile | None =
None, caption: str | None = None,
parse_mode: str |
~aiogram.client.default.Default | None =
<Default('parse_mode')>, caption_entities:
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: InputFileUnion

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.

```
caption_entities: list[MessageEntity] | None
```

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

duration: int | None

Optional. Duration of the audio in seconds

performer: str | None

Optional. Performer of the audio

title: str | None

Optional. Title of the audio

InputMediaDocument

```
class aiogram.types.input_media_document.InputMediaDocument(*, type: ~typ-
                                                                         ing.Literal[InputMediaType.DOCUMENT]
                                                                         = InputMediaType.DOCUMENT,
                                                                         media: str |
                                                                         ~aiogram.types.input_file.InputFile,
                                                                         thumbnail:
                                                                         ~aiogram.types.input file.InputFile |
                                                                         None = None, caption: str | None =
                                                                         None, parse_mode: str |
                                                                         ~aiogram.client.default.Default | None
                                                                         = < Default('parse\_mode')>,
                                                                         caption entities:
                                                                         list[~aiogram.types.message entity.MessageEntity]
                                                                         | None = None,
                                                                         disable_content_type_detection: bool
                                                                         | None = None, **extra\_data:
```

~typing.Any)

Represents a general file to be sent.

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

type: Literal[InputMediaType.DOCUMENT]

Type of the result, must be document

media: InputFileUnion

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

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

~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:
list[~aiogram.types.message_entity.MessageEntity]
| None = None, show_caption_above_media:
bool | ~aiogram.client.default.Default | None
= <Default('show_caption_above_media')>,
has_spoiler: bool | None = None,
**extra_data: ~typing.Any)

Represents a photo to be sent.

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

type: Literal[InputMediaType.PHOTO]

Type of the result, must be photo

media: InputFileUnion

File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file_attach_name>' to upload a new one using multipart/form-data under <file_attach_name> name. More information on Sending Files >>

caption: str | None

Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Optional. Mode for parsing entities in the photo caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

has_spoiler: bool | None

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

InputMediaVideo

class aiogram.types.input_media_video.InputMediaVideo(*, type:

~typing.Literal[InputMediaType.VIDEO] = InputMediaType.VIDEO, media: str | ~aiogram.types.input_file.InputFile, thumbnail: ~aiogram.types.input file.InputFile | None = *None*, *cover*: *str* | ~aiogram.types.input_file.InputFile | None = *None*, *start timestamp:* ~*datetime.datetime* | \sim datetime.timedelta | int | None = None, caption: str | None = None, parse_mode: str | ~aiogram.client.default.Default | None = <*Default('parse_mode')>*, *caption_entities:* list[~aiogram.types.message_entity.MessageEntity] | None = None, show_caption_above_media: bool | ~aiogram.client.default.Default | None = < Default('show_caption_above_media')>, width: int | None = None, height: int | None = *None*, duration: $int \mid None = None$, supports streaming: $bool \mid None = None$, $has_spoiler: bool \mid None = None,$ **extra_data: ~typing.Any)

Represents a video to be sent.

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

type: Literal[InputMediaType.VIDEO]

Type of the result, must be video

media: InputFileUnion

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 >>>

cover: InputFileUnion | None

Optional. Cover for the video in the message. 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 >>

start_timestamp: DateTimeUnion | None

Optional. Start timestamp for the video in the message

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

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

width: int | None

Optional. Video width

height: int | None

Optional. Video height

duration: int | None

Optional. Video duration in seconds

supports_streaming: bool | None

Optional. Pass True if the uploaded video is suitable for streaming

has_spoiler: bool | None

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

InputPaidMedia

class aiogram.types.input_paid_media.InputPaidMedia(**extra_data: Any)

This object describes the paid media to be sent. Currently, it can be one of

- aiogram.types.input_paid_media_photo.InputPaidMediaPhoto
- aiogram.types.input_paid_media_video.InputPaidMediaVideo

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

InputPaidMediaPhoto

class aiogram.types.input_paid_media_photo.InputPaidMediaPhoto(*, type: Lit-

eral[InputPaidMediaType.PHOTO]
= InputPaidMediaType.PHOTO,
media: str | InputFile,
**extra_data: Any)

The paid media to send is a photo.

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

type: Literal[InputPaidMediaType.PHOTO]

Type of the media, must be photo

media: InputFileUnion

File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file_attach_name>' to upload a new one using multipart/form-data under <file_attach_name> name. *More information on Sending Files* »

InputPaidMediaVideo

class aiogram.types.input_paid_media_video.**InputPaidMediaVideo**(*, type: Lit-

eral[InputPaidMediaType.VIDEO]
= InputPaidMediaType.VIDEO,
media: str | InputFile, thumbnail:
InputFile | None = None, cover:
str | InputFile | None = None,
start_timestamp: datetime |
timedelta | int | None = None,
width: int | None = None, height:
int | None = None, duration: int |
None = None,
supports_streaming: bool | None
= None, **extra data: Any)

The paid media to send is a video.

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

type: Literal[InputPaidMediaType.VIDE0]

Type of the media, must be video

media: InputFileUnion

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 "

cover: InputFileUnion | None

Optional. Cover for the video in the message. 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 >>

start_timestamp: DateTimeUnion | None

Optional. Start timestamp for the video in the message

width: int | None

Optional. Video width

height: int | None

Optional. Video height

duration: int | None

Optional. Video duration in seconds

supports_streaming: bool | None

Optional. Pass True if the uploaded video is suitable for streaming

InputPollOption

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

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

text: str

Option text, 1-100 characters

text_parse_mode: str | Default | None

Optional. Mode for parsing entities in the text. See formatting options for more details. Currently, only custom emoji entities are allowed

text_entities: list[MessageEntity] | None

Optional. A JSON-serialized list of special entities that appear in the poll option text. It can be specified instead of *text_parse_mode*

KeyboardButton

class aiogram.types.keyboard_button.KeyboardButton(*, text: str, request_users:

KeyboardButtonRequestUsers | None = None, request_chat: KeyboardButtonRequestChat | None = None, request_contact: bool | None = None, request_location: bool | None = None, request_poll: KeyboardButtonPollType | None = None, web_app: WebAppInfo | None = None, request_user: KeyboardButtonRequestUser | None = None, **extra_data: Any)

This object represents one button of the reply keyboard. At most one of the optional fields must be used to specify type of the button. For simple text buttons, *String* can be used instead of this object to specify the button text. **Note:** request_users and request_chat options will only work in Telegram versions released after 3 February, 2023. Older clients will display unsupported message.

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

text: str

Text of the button. If none of the optional fields are used, it will be sent as a message when the button is pressed

request_users: KeyboardButtonRequestUsers | None

Optional. If specified, pressing the button will open a list of suitable users. Identifiers of selected users will be sent to the bot in a 'users_shared' service message. Available in private chats only.

request_chat: KeyboardButtonRequestChat | None

Optional. If specified, pressing the button will open a list of suitable chats. Tapping on a chat will send its identifier to the bot in a 'chat_shared' service message. Available in private chats only.

request_contact: bool | None

Optional. If True, the user's phone number will be sent as a contact when the button is pressed. Available in private chats only.

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.

KeyboardButtonRequestChat

```
class aiogram.types.keyboard_button_request_chat.KeyboardButtonRequestChat(*, request_id: int,
```

chat_is_channel:

bool,

chat_is_forum:

bool | None =

None,

chat_has_username:

bool | None =

None,

chat_is_created:

bool | None =

None,

user_administrator_rights:

ChatAdministra-

torRights | None =

None,

bot_administrator_rights:

ChatAdministra-

torRights | None =

None,

bot_is_member:

bool | None =

None,

request_title: bool

| None = None, request username:

bool | None =

None,

request_photo:

bool | None =

None,

 $**extra_data:$

Any)

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.

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.

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 | 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.

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
                                                                           = <De-
                                                                          fault('link_preview_is_disabled')>,
                                                                          url: str \mid None = None,
                                                                          prefer small media: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = < De-
                                                                          fault('link_preview_prefer_small_media')>,
                                                                          prefer_large_media: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = <De-
                                                                          fault('link preview prefer large media')>,
                                                                          show_above_text: bool |
                                                                           ~aiogram.client.default.Default | None
                                                                           = <De-
                                                                          fault('link_preview_show_above_text')>,
                                                                           **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

```
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 the sender

longitude: float

Longitude as defined by the sender

horizontal_accuracy: float | None

Optional. The radius of uncertainty for the location, measured in meters; 0-1500

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.

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

MenuButton

```
class aiogram.types.menu_button.MenuButton(*, type: str, text: str \mid None = None, web_app: WebAppInfo \mid None = None, **extra_data: Any)
```

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

```
web_app: WebAppInfo | None
```

Optional. Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method <code>aiogram.methods.answer_web_app_query.AnswerWebAppQuery</code>. Alternatively, a <code>t.me</code> link to a Web App of the bot can be specified in the object instead of the Web App's URL, in which case the Web App will be opened as if the user pressed the link.

MenuButtonCommands

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

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

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

web_app: WebAppInfo

Description of the Web App that will be launched when the user presses the button. The Web App will be able to send an arbitrary message on behalf of the user using the method <code>aiogram.methods.answer_web_app_query.AnswerWebAppQuery</code>. Alternatively, a t.me link to a Web App of the bot can be specified in the object instead of the Web App's URL, in which case the Web App will be opened as if the user pressed the link.

Message

class alogram.types.message.**Message**(*, message id: int, date: datetime serializer, return type=int, when_used=unless - none)], chat: Chat, message_thread_id: int | None = None, from user: User | None = None, sender chat: Chat | $None = None, sender_boost_count: int | None = None,$ *sender_business_bot*: User | *None* = *None*, *business_connection_id*: str | None = None, forward origin: MessageOriginUser | MessageOriginHiddenUser | MessageOriginChat | MessageOriginChannel | None = None, is topic message: bool | None = None, is automatic forward: bool | None = None, reply_to_message: Message | None = None, external_reply: ExternalReplyInfo | None = None, quote: TextQuote | None = None, reply_to_story: Story | None = None, via_bot: User | None = None, edit_date: int | None = None, has_protected_content: bool | None = *None*, *is_from_offline*: *bool* | *None* = *None*, *media_group_id*: *str* | None = None, author_signature: str | None = None, text: str | None = *None*, *entities*: list[MessageEntity] | None = None, *link_preview_options:* LinkPreviewOptions | *None = None, effect_id:* $str \mid None = None$, animation: Animation | None = None, audio: Audio | None = None, document: Document | None = None, paid media: PaidMediaInfo | None = None, photo: list[PhotoSize] | *None* = *None*, *sticker*: Sticker | *None* = *None*, *story*: Story | *None* = None, video: Video | None = None, video note: VideoNote | None = *None*, *voice*: Voice | None = None, caption: str | None = None, caption entities: list/MessageEntity/ | None = None, show caption above media: $bool \mid None = None$, has media spoiler: bool | None = None, contact: Contact | None = *None, dice:* Dice | None = None, game: Game | None = None, poll: Poll | None = None, venue: Venue | None = None, location: Location $| None = None, new_chat_members: list[User] | None = None,$ *left_chat_member:* User | None = None, new_chat_title: str | None = *None*, new_chat_photo : list[PhotoSize] | None = None, delete_chat_photo: bool | None = None, group_chat_created: bool | $None = None, supergroup_chat_created: bool | None = None,$ $channel_chat_created: bool \mid None = None,$ message auto delete timer changed: MessageAutoDeleteTimerChanged | None = None, migrate to chat id: int | None = None, migrate from chat id: int | *None* = *None*, *pinned_message*: Message | InaccessibleMessage | *None* = *None*, *invoice*: Invoice | *None* = *None*, *successful* payment: SuccessfulPayment | *None* = *None*, *refunded_payment*: RefundedPayment | None = None, users shared: UsersShared | None = None, chat shared: ChatShared | None = None, connected website: str | None = None, write access allowed: WriteAccessAllowed | $None = None, passport_data: PassportData | None = None,$ *proximity_alert_triggered:* ProximityAlertTriggered | *None* = *None*, boost_added: ChatBoostAdded | None = None, chat_background_set: ChatBackground | *None* = *None*, *forum_topic_created*: ForumTopicCreated | *None* = *None*, *forum_topic_edited*: ForumTopicEdited | *None* = *None*, *forum_topic_closed*: ForumTopicClosed | *None* = *None*, *forum_topic_reopened*: ForumTopicReopened | *None* = *None*, *general_forum_topic_hidden*: GeneralForumTopicHidden | None = None, general_forum_topic_unhidden: GeneralForumTopicUnhidden | None = None, giveaway created: GiveawayCreated | None = None, *giveaway:* Giveaway | *None* = *None*, *giveaway_winners:* Giveaway Winners | None = None, giveaway completed: 200 GiveawayCompleted | None = None, video_c/Chapten/2ed/Contents

> VideoChatScheduled | None = None, video chat started: VideoChatStarted | None = None, video chat ended:

VideoChatEnded | None = None, video chat participants invited:

This object represents a message.

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

message_id: int

Unique message identifier inside this chat. In specific instances (e.g., message containing a video sent to a big chat), the server might automatically schedule a message instead of sending it immediately. In such cases, this field will be 0 and the relevant message will be unusable until it is actually sent

date: DateTime

Date the message was sent in Unix time. It is always a positive number, representing a valid date.

chat: Chat

Chat the message belongs to

message_thread_id: int | None

Optional. Unique identifier of a message thread to which the message belongs; for supergroups only

from_user: User | None

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

sender_chat: Chat | None

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

sender_boost_count: int | None

Optional. If the sender of the message boosted the chat, the number of boosts added by the user

sender_business_bot: User | None

Optional. The bot that actually sent the message on behalf of the business account. Available only for outgoing messages sent on behalf of the connected business account.

business_connection_id: str | None

Optional. Unique identifier of the business connection from which the message was received. If non-empty, the message belongs to a chat of the corresponding business account that is independent from any potential bot chat which might share the same identifier.

forward_origin: MessageOriginUnion | None

Optional. Information about the original message for forwarded messages

is_topic_message: bool | None

Optional. True, if the message is sent to a forum topic

is_automatic_forward: bool | None

Optional. True, if the message is a channel post that was automatically forwarded to the connected discussion group

reply_to_message: Message | None

Optional. For replies in the same chat and message thread, the original message. Note that the Message object in this field will not contain further *reply_to_message* fields even if it itself is a reply.

external_reply: ExternalReplyInfo | None

Optional. Information about the message that is being replied to, which may come from another chat or forum topic

quote: TextQuote | None Optional. For replies that quote part of the original message, the quoted part of the message reply_to_story: Story | None Optional. For replies to a story, the original story via_bot: User | None Optional. Bot through which the message was sent edit_date: int | None Optional. Date the message was last edited in Unix time has_protected_content: bool | None Optional. True, if the message can't be forwarded is_from_offline: bool | None Optional. True, if the message was sent by an implicit action, for example, as an away or a greeting business message, or as a scheduled message media_group_id: str | None Optional. The unique identifier of a media message group this message belongs to author_signature: str | None Optional. Signature of the post author for messages in channels, or the custom title of an anonymous group administrator text: str | None Optional. For text messages, the actual UTF-8 text of the message entities: list[MessageEntity] | None Optional. For text messages, special entities like usernames, URLs, bot commands, etc. that appear in the text link_preview_options: LinkPreviewOptions | None Optional. Options used for link preview generation for the message, if it is a text message and link preview options were changed effect_id: str | None Optional. Unique identifier of the message effect added to the message animation: Animation | None Optional. Message is an animation, information about the animation. For backward compatibility, when this field is set, the document field will also be set audio: Audio | None Optional. Message is an audio file, information about the file document: Document | None Optional. Message is a general file, information about the file paid_media: PaidMediaInfo | None Optional. Message contains paid media; information about the paid media photo: list[PhotoSize] | None Optional. Message is a photo, available sizes of the photo sticker: Sticker | None

Optional. Message is a sticker, information about the sticker

```
story: Story | None
     Optional. Message is a forwarded story
video: Video | None
     Optional. Message is a video, information about the video
video_note: VideoNote | None
     Optional. Message is a video note, information about the video message
voice: Voice | None
     Optional. Message is a voice message, information about the file
caption: str | None
     Optional. Caption for the animation, audio, document, paid media, photo, video or voice
caption_entities: list[MessageEntity] | None
     Optional. For messages with a caption, special entities like usernames, URLs, bot commands, etc. that
     appear in the caption
show_caption_above_media: bool | None
     Optional. True, if the caption must be shown above the message media
has_media_spoiler: bool | None
     Optional. True, if the message media is covered by a spoiler animation
contact: Contact | None
     Optional. Message is a shared contact, information about the contact
dice: Dice | None
     Optional. Message is a dice with random value
game: Game | None
     Optional. Message is a game, information about the game. More about games »
poll: Poll | None
     Optional. Message is a native poll, information about the poll
venue: Venue | None
     Optional. Message is a venue, information about the venue. For backward compatibility, when this field is
     set, the location field will also be set
location: Location | None
     Optional. Message is a shared location, information about the location
new_chat_members: list[User] | None
     Optional. New members that were added to the group or supergroup and information about them (the bot
     itself may be one of these members)
left_chat_member: User | None
     Optional. A member was removed from the group, information about them (this member may be the bot
     itself)
new_chat_title: str | None
     Optional. A chat title was changed to this value
new_chat_photo: list[PhotoSize] | None
     Optional. A chat photo was change to this value
```

delete_chat_photo: bool | None

Optional. Service message: the chat photo was deleted

group_chat_created: bool | None

Optional. Service message: the group has been created

supergroup_chat_created: bool | None

Optional. Service message: the supergroup has been created. This field can't be received in a message coming through updates, because bot can't be a member of a supergroup when it is created. It can only be found in reply_to_message if someone replies to a very first message in a directly created supergroup.

channel_chat_created: bool | None

Optional. Service message: the channel has been created. This field can't be received in a message coming through updates, because bot can't be a member of a channel when it is created. It can only be found in reply_to_message if someone replies to a very first message in a channel.

message_auto_delete_timer_changed: MessageAutoDeleteTimerChanged | None

Optional. Service message: auto-delete timer settings changed in the chat

migrate_to_chat_id: int | None

Optional. The group has been migrated to a supergroup with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

migrate_from_chat_id: int | None

Optional. The supergroup has been migrated from a group with the specified identifier. This number may have more than 32 significant bits and some programming languages may have difficulty/silent defects in interpreting it. But it has at most 52 significant bits, so a signed 64-bit integer or double-precision float type are safe for storing this identifier.

pinned_message: MaybeInaccessibleMessageUnion | None

Optional. Specified message was pinned. Note that the Message object in this field will not contain further *reply_to_message* fields even if it itself is a reply.

invoice: Invoice | None

Optional. Message is an invoice for a payment, information about the invoice. More about payments »

successful_payment: SuccessfulPayment | None

Optional. Message is a service message about a successful payment, information about the payment. More about payments »

refunded_payment: RefundedPayment | None

Optional. Message is a service message about a refunded payment, information about the payment. More about payments »

users_shared: UsersShared | None

Optional. Service message: users were shared with the bot

chat_shared: ChatShared | None

Optional. Service message: a chat was shared with the bot

connected_website: str | None

Optional. The domain name of the website on which the user has logged in. More about Telegram Login »

```
write_access_allowed: WriteAccessAllowed | None
    Optional. Service message: the user allowed the bot to write messages after adding it to the attachment or
    side menu, launching a Web App from a link, or accepting an explicit request from a Web App sent by the
    method requestWriteAccess
passport_data: PassportData | None
    Optional. Telegram Passport data
proximity_alert_triggered: ProximityAlertTriggered | None
    Optional. Service message. A user in the chat triggered another user's proximity alert while sharing Live
    Location.
boost_added: ChatBoostAdded | None
    Optional. Service message: user boosted the chat
chat_background_set: ChatBackground | None
    Optional. Service message: chat background set
forum_topic_created: ForumTopicCreated | None
    Optional. Service message: forum topic created
forum_topic_edited: ForumTopicEdited | None
    Optional. Service message: forum topic edited
forum_topic_closed: ForumTopicClosed | None
    Optional. Service message: forum topic closed
forum_topic_reopened: ForumTopicReopened | None
    Optional. Service message: forum topic reopened
general_forum_topic_hidden: GeneralForumTopicHidden | None
    Optional. Service message: the 'General' forum topic hidden
general_forum_topic_unhidden: GeneralForumTopicUnhidden | None
    Optional. Service message: the 'General' forum topic unhidden
giveaway_created: GiveawayCreated | None
    Optional. Service message: a scheduled giveaway was created
giveaway: Giveaway | None
    Optional. The message is a scheduled giveaway message
giveaway_winners: GiveawayWinners | None
    Optional. A giveaway with public winners was completed
giveaway_completed: GiveawayCompleted | None
    Optional. Service message: a giveaway without public winners was completed
video_chat_scheduled: VideoChatScheduled | None
    Optional. Service message: video chat scheduled
video_chat_started: VideoChatStarted | None
    Optional. Service message: video chat started
video_chat_ended: VideoChatEnded | None
    Optional. Service message: video chat ended
```

video_chat_participants_invited: VideoChatParticipantsInvited | None

Optional. Service message: new participants invited to a video chat

web_app_data: WebAppData | None

Optional. Service message: data sent by a Web App

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message. login_url buttons are represented as ordinary url buttons.

forward_date: DateTime | None

Optional. For forwarded messages, date the original message was sent in Unix time

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

forward_from: User | None

Optional. For forwarded messages, sender of the original message

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

forward_from_chat: Chat | None

Optional. For messages forwarded from channels or from anonymous administrators, information about the original sender chat

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

forward_from_message_id: int | None

Optional. For messages forwarded from channels, identifier of the original message in the channel

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

forward_sender_name: str | None

Optional. Sender's name for messages forwarded from users who disallow adding a link to their account in forwarded messages

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

forward_signature: str | None

Optional. For forwarded messages that were originally sent in channels or by an anonymous chat 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

```
 \textbf{as\_reply\_parameters}(allow\_sending\_without\_reply: Optional[Union[bool, Default]] = \\ < Default('allow\_sending\_without\_reply')>, quote: Optional[str] = None, \\ quote\_parse\_mode: Optional[Union[str, Default]] = < Default('parse\_mode')>, \\ quote\_entities: Optional[List[MessageEntity]] = None, quote\_position: \\ Optional[int] = None) \rightarrow ReplyParameters
```

```
reply_animation(animation: InputFileUnion, duration: Optional[int] = None, width: Optional[int] = None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = 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_parameters

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent aiogram.types.message.Message is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- animation Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data. More information on Sending Files »
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation

- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_animation.SendAnimation

answer_animation (animation: InputFileUnion, duration: Optional[int] = None, width: Optional[int] =

None, height: Optional[int] = None, thumbnail: Optional[InputFile] = None, caption:

Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None,

show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,

disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool,

Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] =

None, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] =

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 <code>aiogram.types.message</code> is returned. Bots can currently send animation files of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- animation Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data. More information on Sending Files »
- duration Duration of sent animation in seconds
- width Animation width
- height Animation height

- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **caption** Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the animation caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the animation needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_animation.SendAnimation

reply_audio(audio: InputFileUnion, 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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendAudio

Shortcut for method aiogram.methods.send_audio.SendAudio will automatically fill method attributes:

- chat_id
- message_thread_id

- business_connection_id
- reply_parameters

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

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

Parameters

- audio Audio file to send. Pass a file_id as String to send an audio file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an audio file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- caption Audio caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- duration Duration of the audio in seconds
- **performer** Performer
- title Track name
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_audio.SendAudio

answer_audio(audio: InputFileUnion, 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') >, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_audio.SendAudio

 $\begin{aligned} \textbf{reply_contact}(phone_number: str, first_name: str, last_name: Optional[str] = None, vcard: Optional[str] \\ &= None, disable_notification: Optional[bool] = None, protect_content: \\ &Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: \\ &Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: \\ &Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = \\ &None, **kwargs: Any) \rightarrow SendContact \end{aligned}$

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

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

• allow_sending_without_reply - Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_contact.SendContact

answer_contact(phone_number: str, first_name: str, last_name: Optional[str] = None, vcard:

Optional[str] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast:

Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] =

None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id:

Optional[int] = None, **kwargs: Any) → SendContact

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

- chat_id
- message_thread_id
- business_connection_id

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

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

Parameters

- **phone_number** Contact's phone number
- **first_name** Contact's first name
- last_name Contact's last name
- vcard Additional data about the contact in the form of a vCard, 0-2048 bytes
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_contact.SendContact

 $\begin{aligned} \textbf{reply_document:} & \textit{InputFileUnion, thumbnail: Optional[InputFile]} = \textit{None, caption: Optional[str]} \\ &= \textit{None, parse_mode: Optional[Union[str, Default]]} = \textit{Obefault('parse_mode')>}, \\ &caption_entities: Optional[list[MessageEntity]] = \textit{None, disable_content_type_detection:} \\ &Optional[bool] = \textit{None, disable_notification: Optional[bool]} = \textit{None, protect_content:} \\ &Optional[Union[bool, Default]] = \textit{Obefault('protect_content')>}, allow_paid_broadcast:} \\ &Optional[bool] = \textit{None, message_effect_id: Optional[str]} = \textit{None, reply_markup:} \\ &Optional[ReplyMarkupUnion] = \textit{None, allow_sending_without_reply: Optional[bool]} = \\ &\textit{None, **kwargs: Any)} \rightarrow \textit{SendDocument} \end{aligned}$

 $Shortcut\ for\ method\ aiogram. {\it methods.send_document}. Send Document\ will\ automatically\ fill\ method\ attributes:$

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_document.SendDocument

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_document.SendDocument

 $\textbf{reply_game}(game_short_name: str, disable_notification: Optional[bool] = None, protect_content: \\ Optional[Union[bool, Default]] = & Optional['protect_content') >, allow_paid_broadcast: \\ Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: \\ Optional[InlineKeyboardMarkup] = None, allow_sending_without_reply: Optional[bool] = \\ None, **kwargs: Any) \rightarrow SendGame$

Shortcut for method aiogram.methods.send_game.SendGame will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only

- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_game.SendGame

answer_game($game_short_name$: str, $disable_notification$: Optional[bool] = None, $protect_content$: $Optional[Union[bool, Default]] = <Default('protect_content')>$, $allow_paid_broadcast$: Optional[bool] = None, $message_effect_id$: Optional[str] = None, $reply_parameters$: Optional[ReplyParameters] = None, $reply_markup$: Optional[InlineKeyboardMarkup] = None, $allow_sending_without_reply$: Optional[bool] = None, $reply_to_message_id$: optional[int] = None, **skwargs: skwargs: skwargs

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_game.SendGame

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

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- payload Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- prices Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **provider_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent

- message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider_data JSON-serialized data about the invoice, which will be shared with the
 payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo_size** Photo size in bytes
- photo_width Photo width
- photo_height Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need_shipping_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_invoice.SendInvoice

Shortcut for method <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

- title Product name, 1-32 characters
- **description** Product description, 1-255 characters
- **payload** Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use it for your internal processes.
- **currency** Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.
- prices Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.
- **provider_token** Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.
- max_tip_amount The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.
- **suggested_tip_amounts** A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.
- **start_parameter** Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent

- message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter
- provider_data JSON-serialized data about the invoice, which will be shared with the
 payment provider. A detailed description of required fields should be provided by the payment provider.
- **photo_url** URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.
- **photo_size** Photo size in bytes
- photo_width Photo width
- photo_height Photo height
- **need_name** Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.
- **need_phone_number** Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.
- need_email Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.
- **need_shipping_address** Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.
- **send_phone_number_to_provider** Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.
- **send_email_to_provider** Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.
- **is_flexible** Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_parameters** Description of the message to reply to
- **reply_markup** A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_invoice.SendInvoice

reply_location(latitude: float, longitude: float, horizontal_accuracy: Optional[float] = None, live_period:

Optional[int] = None, heading: Optional[int] = None, proximity_alert_radius:

Optional[int] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast:

Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup:

Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendLocation

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

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

Use this method to send point on the map. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- **latitude** Latitude of the location
- longitude Longitude of the location
- horizontal_accuracy The radius of uncertainty for the location, measured in meters;
 0-1500
- **live_period** Period in seconds during which the location will be updated (see Live Locations, should be between 60 and 86400, or 0x7FFFFFFF for live locations that can be edited indefinitely.
- **heading** For live locations, a direction in which the user is moving, in degrees. Must be between 1 and 360 if specified.
- **proximity_alert_radius** For live locations, a maximum distance for proximity alerts about approaching another chat member, in meters. Must be between 1 and 100000 if specified.
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_location.SendLocation

answer_location(latitude: float, longitude: float, horizontal_accuracy: Optional[float] = None, live_period: Optional[int] = None, heading: Optional[int] = None, proximity_alert_radius: Optional[int] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_location.SendLocation

 $\begin{aligned} \textbf{reply_media_group}(\textit{media: list[MediaUnion], disable_notification: Optional[bool]} &= \textit{None,} \\ \textit{protect_content: Optional[Union[bool, Default]]} &= < \textit{Default('protect_content')} >, \\ \textit{allow_paid_broadcast: Optional[bool]} &= \textit{None, message_effect_id: Optional[str]} &= \\ \textit{None, allow_sending_without_reply: Optional[bool]} &= \textit{None, **kwargs: Any}) \rightarrow \\ \textit{SendMediaGroup} \end{aligned}$

Shortcut for method <code>aiogram.methods.send_media_group.SendMediaGroup</code> will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

Parameters

- media A JSON-serialized array describing messages to be sent, must include 2-10 items
- disable_notification Sends messages silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent messages from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

 $instance\ of\ method\ aiogram.methods.send_media_group.Send\\ MediaGroup$

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the messages are a reply, ID of the original message

Returns

instance of method aiogram.methods.send_media_group.SendMediaGroup

```
reply(text: str, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities:

Optional[list[MessageEntity]] = None, link_preview_options: Optional[Union[LinkPreviewOptions,
Default]] = <Default('link_preview')>, disable_notification: Optional[bool] = None, protect_content:
Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast:
Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None,
disable_web_page_preview: Optional[Union[bool, Default]] =
<Default('link_preview_is_disabled')>, **kwargs: Any) → SendMessage
```

Shortcut for method $aiogram.methods.send_message.SendMessage$ will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message

instance of method aiogram.methods.send_message.SendMessage

```
answer(text: str, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, entities:

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

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

- chat_id
- message_thread_id
- business_connection_id

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

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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- disable_web_page_preview Disables link previews for links in this message
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_message.SendMessage

```
reply_photo(photo: InputFileUnion, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendPhoto
```

Shortcut for method aiogram.methods.send_photo.SendPhoto will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

Parameters

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.

- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_photo.SendPhoto

answer_photo(photo: InputFileUnion, caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendPhoto

Shortcut for method <code>aiogram.methods.send_photo.SendPhoto</code> will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id

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

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

- **photo** Photo to send. Pass a file_id as String to send a photo that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a photo from the Internet, or upload a new photo using multipart/form-data. The photo must be at most 10 MB in size. The photo's width and height must not exceed 10000 in total. Width and height ratio must be at most 20. *More information on Sending Files* »
- **caption** Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters after entities parsing

- parse_mode Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the photo needs to be covered with a spoiler animation
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_photo.SendPhoto

reply_poll(question: str, options: list[InputPollOptionUnion], 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[DateTimeUnion] = None, is_closed: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendPoll

Shortcut for method aiogram.methods.send_poll.SendPoll will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_poll.SendPoll

answer_poll(question: str, options: list[InputPollOptionUnion], 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[DateTimeUnion] = None, is_closed: Optional[bool] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_poll.SendPoll

Shortcut for method <code>aiogram.methods.send_dice.SendDice</code> will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

- **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

- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_dice.SendDice

answer_dice(emoji: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content:

Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast:

Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters:

Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_dice.SendDice

reply_sticker(sticker: InputFileUnion, emoji: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendSticker

Shortcut for method aiogram.methods.send_sticker.SendSticker will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- **sticker** Sticker to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a .WEBP sticker from the Internet, or upload a new .WEBP, .TGS, or .WEBM sticker using multipart/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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_sticker.SendSticker

answer_sticker(sticker: InputFileUnion, emoji: Optional[str] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_sticker.SendSticker

reply_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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVenue

Shortcut for method aiogram.methods.send_venue.SendVenue will automatically fill method attributes:

- chat_id
- message_thread_id

- business_connection_id
- reply_parameters

Use this method to send information about a venue. On success, the sent aiogram.types.message. Message is returned.

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

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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_venue.SendVenue

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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

Returns

instance of method aiogram.methods.send_venue.SendVenue

reply_video(video: InputFileUnion, duration: Optional[int] = None, width: Optional[int] = None, height:

Optional[int] = None, thumbnail: Optional[InputFile] = None, cover:

Optional[InputFileUnion] = None, start_timestamp: Optional[DateTimeUnion] = None,
caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_markup: Optional[ReplyMarkupUnion] = 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_parameters

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as aiogram.types.document.Document). On success, the sent aiogram.types.message.Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

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

- **video** Video to send. Pass a file_id as String to send a video that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a video from the Internet, or upload a new video using multipart/form-data. *More information on Sending Files* »
- duration Duration of sent video in seconds
- width Video width
- height Video height
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*"""
- **cover** Cover for the video in the message. 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* »
- **start_timestamp** Start timestamp for the video in the message
- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- show_caption_above_media Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- supports_streaming Pass True if the uploaded video is suitable for streaming

- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

instance of method aiogram.methods.send_video.SendVideo

```
answer_video(video: InputFileUnion, duration: Optional[int] = None, width: Optional[int] = None, height:

Optional[int] = None, thumbnail: Optional[InputFile] = None, cover:

Optional[InputFileUnion] = None, start_timestamp: Optional[DateTimeUnion] = None,
caption: Optional[str] = None, parse_mode: Optional[Union[str, Default]] =

<Default('parse_mode')>, caption_entities: Optional[list[MessageEntity]] = None,
show_caption_above_media: Optional[Union[bool, Default]] =

<Default('show_caption_above_media')>, has_spoiler: Optional[bool] = None,
supports_streaming: Optional[bool] = None, disable_notification: Optional[bool] = None,
protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,
allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None,
reply_parameters: Optional[ReplyParameters] = None, reply_markup:
Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] =
None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → SendVideo
```

Shortcut for method aiogram.methods.send_video.SendVideo will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id

Use this method to send video files, Telegram clients support MPEG4 videos (other formats may be sent as 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*"""
- **cover** Cover for the video in the message. 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* »
- **start_timestamp** Start timestamp for the video in the message
- **caption** Video caption (may also be used when resending videos by *file_id*), 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the video caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- has_spoiler Pass True if the video needs to be covered with a spoiler animation
- **supports_streaming** Pass True if the uploaded video is suitable for streaming
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_video.SendVideo

```
reply_video_note (video_note: InputFileUnion, duration: Optional[int] = None, length: Optional[int] = None, thumbnail: Optional[InputFile] = None, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVideoNote
```

Shortcut for method aiogram.methods.send_video_note.SendVideoNote will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- video_note Video note to send. Pass a file_id as String to send a video note that exists
 on the Telegram servers (recommended) or upload a new video using multipart/form-data.
 More information on Sending Files ». Sending video notes by a URL is currently unsupported
- duration Duration of sent video in seconds
- length Video width and height, i.e. diameter of the video message
- **thumbnail** Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files*
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_video_note.SendVideoNote

answer_video_note(video_note: InputFileUnion, duration: Optional[int] = None, length: Optional[int] =
None, thumbnail: Optional[InputFile] = None, disable_notification: Optional[bool] =
None, protect_content: Optional[Union[bool, Default]] =
<Default('protect_content')>, allow_paid_broadcast: Optional[bool] = None,
message_effect_id: Optional[str] = None, reply_parameters:
Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] =
None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id:
Optional[int] = None, **kwargs: Any) → 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

- 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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_video_note.SendVideoNote

reply_voice(voice: InputFileUnion, 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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, **kwargs: Any) → SendVoice

Shortcut for method aiogram.methods.send_voice.SendVoice will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

Parameters

- **voice** Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »
- caption Voice message caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the voice message caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **duration** Duration of the voice message in seconds
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **message_effect_id** Unique identifier of the message effect to be added to the message; for private chats only
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

• allow_sending_without_reply - Pass True if the message should be sent even if the specified replied-to message is not found

Returns

instance of method aiogram.methods.send_voice.SendVoice

answer_voice(voice: InputFileUnion, 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')>, allow_paid_broadcast: Optional[bool] = None, message_effect_id: Optional[str] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = 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</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

- **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
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- message_effect_id Unique identifier of the message effect to be added to the message; for private chats only
- reply_parameters Description of the message to reply to

- reply_markup Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user
- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.send_voice.SendVoice

send_copy (chat_id: str | int, disable_notification: bool | None = None, reply_to_message_id: int | None = None, reply_parameters: ReplyParameters | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | None = None, allow_sending_without_reply: bool | None = None, message_thread_id: int | None = None, business_connection_id: str | None = None, parse_mode: $str \mid None = None, message_effect_id: str \mid None = None) \rightarrow ForwardMessage \mid SendAnimation \mid$ SendAudio | SendContact | SendDocument | SendLocation | SendMessage | SendPhoto | SendPoll | SendDice | SendSticker | SendVenue | SendVideo | SendVideoNote | SendVoice

Send copy of a message.

Is similar to aiogram.client.bot.Bot.copy_message() but returning the sent message instead of aiogram.types.message_id.MessageId



1 Note

This method doesn't use the API method named copyMessage and historically implemented before the similar method is added to API

Parameters

- · chat_id
- disable_notification
- reply_to_message_id
- reply_parameters
- reply_markup
- allow_sending_without_reply
- message_thread_id
- · business_connection_id
- parse_mode
- message_effect_id

Returns

copy_to(chat_id: ChatIdUnion, message_thread_id: Optional[int] = None, video_start_timestamp:

Optional[DateTimeUnion] = None, caption: Optional[str] = None, parse_mode:

Optional[Union[str, Default]] = <Default('parse_mode')>, caption_entities:

Optional[list[MessageEntity]] = None, show_caption_above_media: Optional[Union[bool, Default]] = <Default('show_caption_above_media')>, disable_notification: Optional[bool] = None, protect_content: Optional[Union[bool, Default]] = <Default('protect_content')>,

allow_paid_broadcast: Optional[bool] = None, reply_parameters: Optional[ReplyParameters] = None, reply_markup: Optional[ReplyMarkupUnion] = None, allow_sending_without_reply: Optional[bool] = None, reply_to_message_id: Optional[int] = None, **kwargs: Any) → CopyMessage

Shortcut for method <code>aiogram.methods.copy_message.CopyMessage</code> will automatically fill method attributes:

- from_chat_id
- message_id

Use this method to copy messages of any kind. Service messages, paid media messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz aiogram.methods. poll.Poll can be copied only if the value of the field <code>correct_option_id</code> is known to the bot. The method is analogous to the method <code>aiogram.methods.forward_message.ForwardMessage</code>, but the copied message doesn't have a link to the original message. Returns the <code>aiogram.types.message_id.MessageId</code> of the sent message on success.

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

- **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
- video_start_timestamp New start timestamp for the copied video in the message
- **caption** New caption for media, 0-1024 characters after entities parsing. If not specified, the original caption is kept
- parse_mode Mode for parsing entities in the new caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the new caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media. Ignored if a new caption isn't specified.
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- reply_parameters Description of the message to reply to
- **reply_markup** Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

- allow_sending_without_reply Pass True if the message should be sent even if the specified replied-to message is not found
- reply_to_message_id If the message is a reply, ID of the original message

instance of method aiogram.methods.copy_message.CopyMessage

```
edit_text (text: str, inline_message_id: Optional[str] = None, parse_mode: Optional[Union[str, Default]]
= <Default('parse_mode')>, entities: Optional[list[MessageEntity]] = None,
link_preview_options: Optional[Union[LinkPreviewOptions, Default]] =
<Default('link_preview')>, reply_markup: Optional[InlineKeyboardMarkup] = None,
disable_web_page_preview: Optional[Union[bool, Default]] =
<Default('link_preview_is_disabled')>, **kwargs: Any) → EditMessageText
```

Shortcut for method <code>aiogram.methods.edit_message_text.EditMessageText</code> will automatically fill method attributes:

- chat id
- message_id
- business_connection_id

Use this method to edit text and game messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message</code> is returned, otherwise <code>True</code> is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within <code>48</code> hours from the time they were sent.

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

Parameters

- text New text of the message, 1-4096 characters after entities parsing
- **inline_message_id** Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
- parse_mode Mode for parsing entities in the message text. See formatting options for more details.
- **entities** A JSON-serialized list of special entities that appear in message text, which can be specified instead of *parse_mode*
- link_preview_options Link preview generation options for the message
- **reply_markup** A JSON-serialized object for an inline keyboard.
- disable_web_page_preview Disables link previews for links in this message

Returns

instance of method aiogram.methods.edit_message_text.EditMessageText

 $\label{lem:content} \textbf{forward}(chat_id: ChatIdUnion, message_thread_id: Optional[int] = None, video_start_timestamp: \\ Optional[DateTimeUnion] = None, disable_notification: Optional[bool] = None, protect_content: \\ Optional[Union[bool, Default]] = <Default('protect_content')>, **kwargs: Any) \rightarrow \\ ForwardMessage$

Shortcut for method aiogram.methods.forward_message.ForwardMessage will automatically fill method attributes:

- from_chat_id
- message_id

Use this method to forward messages of any kind. Service messages and messages with protected content can't be forwarded. On success, the sent aiogram.types.message.Message is returned.

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

Parameters

- **chat_id** Unique identifier for the target chat or username of the target channel (in the format @channelusername)
- **message_thread_id** Unique identifier for the target message thread (topic) of the forum; for forum supergroups only
- video_start_timestamp New start timestamp for the forwarded video in the message
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- **protect_content** Protects the contents of the forwarded message from forwarding and saving

Returns

instance of method aiogram.methods.forward_message.ForwardMessage

edit_media($media: InputMediaUnion, inline_message_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, **kwargs: Any) <math>\rightarrow EditMessageMedia$

Shortcut for method <code>aiogram.methods.edit_message_media.EditMessageMedia</code> will automatically fill method attributes:

- chat_id
- message_id
- business_connection_id

Use this method to edit animation, audio, document, photo, or video messages, or to add media to text messages. If a message is part of a message album, then it can be edited only to an audio for audio albums, only to a document for document albums and to a photo or a video otherwise. When an inline message is edited, a new file can't be uploaded; use a previously uploaded file via its file_id or specify a URL. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within **48 hours** from the time they were sent.

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

Parameters

- media A JSON-serialized object for a new media content of the message
- inline_message_id Required if chat_id and message_id are not specified. Identifier
 of the inline message
- reply_markup A JSON-serialized object for a new inline keyboard.

Returns

instance of method aiogram.methods.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
- business_connection_id

Use this method to edit only the reply markup of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within **48 hours** from the time they were sent.

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

Parameters

- **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 <code>aiogram.methods.edit_message_reply_markup.EditMessageReplyMarkup</code> will automatically fill method attributes:

- chat_id
- message_id
- business_connection_id
- reply_markup

Use this method to edit only the reply markup of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

Parameters

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
```

Shortcut for method aiogram.methods.edit_message_live_location. EditMessageLiveLocation will automatically fill method attributes:

- chat_id
- message_id
- · business_connection_id

Use this method to edit live location messages. A location can be edited until its <code>live_period</code> expires or editing is explicitly disabled by a call to <code>aiogram.methods.stop_message_live_location</code>. StopMessageLiveLocation. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise <code>True</code> is returned.

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

Parameters

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

 $stop_live_location(inline_message_id: str \mid None = None, reply_markup: InlineKeyboardMarkup \mid None = None, **kwargs: Any) <math>\rightarrow StopMessageLiveLocation$

Shortcut for method aiogram.methods.stop_message_live_location. StopMessageLiveLocation will automatically fill method attributes:

- chat_id
- message_id
- business_connection_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.

Returns

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, show_caption_above_media: Optional[Union[bool, Default]] = \langle Default('show\_caption\_above\_media') \rangle, 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
- business_connection_id

Use this method to edit captions of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

Parameters

- **inline_message_id** Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
- caption New caption of the message, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the message caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media. Supported only for animation, photo and video messages.
- reply_markup A JSON-serialized object for an inline keyboard.

Returns

instance of method aiogram.methods.edit_message_caption.EditMessageCaption

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

Shortcut for method $aiogram.methods.delete_message.DeleteMessage$ will automatically fill method attributes:

- chat_id
- message_id
- business_connection_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

 $pin(disable_notification: bool | None = None, **kwargs: Any) \rightarrow PinChatMessage$

Shortcut for method $aiogram.methods.pin_chat_message.PinChatMessage$ will automatically fill method attributes:

- chat_id
- message_id
- business_connection_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

 $instance\ of\ method\ aiogram.methods.pin_chat_message.PinChat\texttt{Message}$

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

Shortcut for method $aiogram.methods.unpin_chat_message.UnpinChatMessage$ will automatically fill method attributes:

- chat_id
- message_id
- business_connection_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($force_private: bool = False, include_thread_id: bool = False) <math>\rightarrow$ str | None

Returns message URL. Cannot be used in private (one-to-one) chats. If chat has a username, returns URL like https://t.me/username/message_id Otherwise (or if {force_private} flag is set), returns https://t.me/c/shifted_chat_id/message_id

Parameters

- **force_private** if set, a private URL is returned even for a public chat
- include_thread_id if set, adds chat thread id to URL and returns like https://t.me/ username/thread_id/message_id

string with full message URL

 $react(reaction: list[ReactionTypeUnion] | None = None, is_big: bool | None = None, **kwargs: Any) \rightarrow SetMessageReaction$

Shortcut for method aiogram.methods.set_message_reaction.SetMessageReaction will automatically fill method attributes:

- · chat id
- message_id
- business_connection_id

Use this method to change the chosen reactions on a message. Service messages of some types can't be reacted to. Automatically forwarded messages from a channel to its discussion group have the same available reactions as messages in the channel. Bots can't use paid reactions. Returns True on success.

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

Parameters

- **reaction** A JSON-serialized list of reaction types to set on the message. Currently, as non-premium users, bots can set up to one reaction per message. A custom emoji reaction can be used if it is either already present on the message or explicitly allowed by chat administrators. Paid reactions can't be used by bots.
- **is_big** Pass True to set the reaction with a big animation

Returns

instance of method aiogram.methods.set_message_reaction.SetMessageReaction

Shortcut for method aiogram.methods.send_paid_media.SendPaidMedia will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id

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

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

Parameters

- **star_count** The number of Telegram Stars that must be paid to buy access to the media; 1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items

- payload Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- caption Media caption, 0-1024 characters after entities parsing
- parse_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- disable_notification Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **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

instance of method aiogram.methods.send_paid_media.SendPaidMedia

```
reply_paid_media(star_count: int, media: list[InputPaidMediaUnion], payload: str | None = None, caption: str | None = None, parse_mode: str | None = None, caption_entities: list[MessageEntity] | None = None, show_caption_above_media: bool | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, allow_paid_broadcast: bool | None = None, reply_markup: ReplyMarkupUnion | None = None, **kwargs: Any) → SendPaidMedia
```

Shortcut for method aiogram.methods.send_paid_media.SendPaidMedia will automatically fill method attributes:

- chat_id
- message_thread_id
- business_connection_id
- reply_parameters

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

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

Parameters

- **star_count** The number of Telegram Stars that must be paid to buy access to the media; 1-2500
- media A JSON-serialized array describing the media to be sent; up to 10 items
- payload Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.
- caption Media caption, 0-1024 characters after entities parsing

- parse_mode Mode for parsing entities in the media caption. See formatting options for more details.
- **caption_entities** A JSON-serialized list of special entities that appear in the caption, which can be specified instead of *parse_mode*
- **show_caption_above_media** Pass True, if the caption must be shown above the message media
- **disable_notification** Sends the message silently. Users will receive a notification with no sound.
- protect_content Protects the contents of the sent message from forwarding and saving
- allow_paid_broadcast Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance
- **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

instance of method aiogram.methods.send_paid_media.SendPaidMedia

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

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 or #hashtag@chatusername), 'cashtag' (\$USD or \$USD@chatusername), 'bot_command' (/start@jobs_bot), 'url' (https://telegram.org), 'email' (do-not-reply@telegram.org), 'phone_number' (+1-212-555-0123), 'bold' (bold text), 'italic' (italic text), 'underline' (underlined

text), 'strikethrough' (strikethrough text), 'spoiler' (spoiler message), 'blockquote' (block quotation), 'expandable_blockquote' (collapsed-by-default block quotation), 'code' (monowidth string), 'pre' (monowidth block), 'text_link' (for clickable text URLs), 'text_mention' (for users without usernames), 'custom_emoji' (for inline custom emoji stickers)

offset: int

Offset in UTF-16 code units to the start of the entity

length: int

Length of the entity in UTF-16 code units

url: str | None

Optional. For 'text_link' only, URL that will be opened after user taps on the text

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. In specific instances (e.g., message containing a video sent to a big chat), the server might automatically schedule a message instead of sending it immediately. In such cases, this field will be 0 and the relevant message will be unusable until it is actually sent

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

MessageOriginChannel

class aiogram.types.message_origin_channel.MessageOriginChannel(*, type: Literal[Messag

eral [Message Origin Type. CHANNEL]

=

MessageOriginType.CHANNEL, date: _datetime_serializer, return_type=int, when_used=unless - none)], chat: Chat, message_id: int, author_signature: str | None = None, **extra_data: Any)

The message was originally sent to a channel chat.

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

type: Literal[MessageOriginType.CHANNEL]

Type of the message origin, always 'channel'

date: DateTime

Date the message was sent originally in Unix time

chat: Chat

Channel chat to which the message was originally sent

message_id: int

Unique message identifier inside the chat

author_signature: str | None

Optional. Signature of the original post author

MessageOriginChat

class aiogram.types.message_origin_chat.MessageOriginChat(*, type:

Literal[MessageOriginType.CHAT] =
MessageOriginType.CHAT, date:
_datetime_serializer, return_type=int,
when_used=unless - none)],
sender_chat: Chat, author_signature: str
| None = None, **extra_data: Any)

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

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

type: Literal[MessageOriginType.CHAT]

Type of the message origin, always 'chat'

date: DateTime

Date the message was sent originally in Unix time

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

 $\textbf{class} \ \, \textbf{aiogram.types.message_origin_hidden_user.} \\ \textbf{MessageOriginHiddenUser}(*, \textit{type: Lit-literate}(*, \textit{type: Literate}(*, \textit{type: Literate}(*, \textit{type: Literate}(*, \textit{type: Literate}(*, \textit{type: Lit-literate}(*, \textit{type: Lit-literate}(*, \textit{type: Lit-literate}(*, \textit{type: Lit-literate}(*, \textit{type: Lit-literate}(*, \textit{type: Lit-literate}(*, \textit{type: Literate}(*, \textit{typ$

eral[MessageOriginType.HIDDEN_USER
= MessageOriginType.HIDDEN_USER,
date:
__datetime_serializer,
return_type=int,
when_used=unless none)],
sender_user_name: str,
**extra_data: Any)

The message was originally sent by an unknown user.

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

type: Literal[MessageOriginType.HIDDEN_USER]

Type of the message origin, always 'hidden_user'

date: _datetime_serializer, return_type=int, when_used=unless-none)]

Date the message was sent originally in Unix time

sender_user_name: str

Name of the user that sent the message originally

MessageOriginUser

class aiogram.types.message_origin_user.MessageOriginUser(*, type:

Literal[MessageOriginType.USER] =
MessageOriginType.USER, date:
_datetime_serializer, return_type=int,
when_used=unless - none)],
sender_user: User, **extra_data: Any)

The message was originally sent by a known user.

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

type: Literal[MessageOriginType.USER]

Type of the message origin, always 'user'

date: DateTime

Date the message was sent originally in Unix time

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_serializer,
return_type=int,
when_used=unless
- none)],
reactions:
list[ReactionCount],
**extra_data:
Any)

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

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

chat: Chat

The chat containing the message

message_id: int

Unique message identifier inside the chat

date: DateTime

Date of the change in Unix time reactions: list[ReactionCount]

List of reactions that are present on the message

MessageReactionUpdated

class aiogram.types.message_reaction_updated.MessageReactionUpdated(*, chat: Chat, message_id:

_datetime_serializer, return_type=int, when_used=unless none)], old_reaction: list[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid], new_reaction: list[ReactionTypeEmoji | ReactionTypeCustomEmoji

int, date:

ReactionTypeCustomEmoji | ReactionTypePaid], user: User | None = None, actor_chat: Chat | None = None, **extra_data: Any)

This object represents a change of a reaction on a message performed by a user.

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

chat: Chat

The chat containing the message the user reacted to

message_id: int

Unique identifier of the message inside the chat

date: DateTime

Date of the change in Unix time

old_reaction: list[ReactionTypeUnion]

Previous list of reaction types that were set by the user

new_reaction: list[ReactionTypeUnion]

New list of reaction types that have been set by the user

user: *User* | None

Optional. The user that changed the reaction, if the user isn't anonymous

actor_chat: Chat | None

Optional. The chat on behalf of which the reaction was changed, if the user is anonymous

PaidMedia

class aiogram.types.paid_media.PaidMedia(**extra_data: Any)

This object describes paid media. Currently, it can be one of

- aiogram.types.paid_media_preview.PaidMediaPreview
- aiogram.types.paid_media_photo.PaidMediaPhoto
- aiogram.types.paid_media_video.PaidMediaVideo

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

PaidMediaInfo

class aiogram.types.paid_media_info.PaidMediaInfo(*, star_count: int, paid_media:

list[PaidMediaPreview | PaidMediaPhoto |
PaidMediaVideo], **extra_data: Any)

Describes the paid media added to a message.

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

star_count: int

The number of Telegram Stars that must be paid to buy access to the media

paid_media: list[PaidMediaUnion]

Information about the paid media

PaidMediaPhoto

The paid media is a photo.

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

type: Literal[PaidMediaType.PHOTO]

Type of the paid media, always 'photo'

photo: list[PhotoSize]

The photo

PaidMediaPreview

The paid media isn't available before the payment.

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

type: Literal[PaidMediaType.PREVIEW]

Type of the paid media, always 'preview'

width: int | None

Optional. Media width as defined by the sender

height: int | None

Optional. Media height as defined by the sender

duration: int | None

Optional. Duration of the media in seconds as defined by the sender

PaidMediaVideo

The paid media is a video.

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

type: Literal[PaidMediaType.VIDEO]

Type of the paid media, always 'video'

video: Video
The video

PhotoSize

class aiogram.types.photo_size.**PhotoSize**(*, file_id: str, file_unique_id: str, width: int, height: int, file_size: int | None = None, **extra_data: Any)

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

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

file_id: str

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

file_unique_id: str

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

width: int

Photo width

height: int

Photo height

file_size: int | None

Optional. File size in bytes

Poll

This object contains information about a poll.

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

id: str

Unique poll identifier

question: str

Poll question, 1-300 characters

options: list[PollOption]

List of poll options

total_voter_count: int

Total number of users that voted in the poll

is_closed: bool

True, if the poll is closed

is_anonymous: bool

True, if the poll is anonymous

type: str

Poll type, currently can be 'regular' or 'quiz'

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.

voter_chat: Chat | None

Optional. The chat that changed the answer to the poll, if the voter is anonymous

user: User | None

Optional. The user that changed the answer to the poll, if the voter isn't anonymous

PollOption

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

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

text: str

Option text, 1-100 characters

voter_count: int

Number of users that voted for this option

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

distance: int

The distance between the users

ReactionCount

Represents a reaction added to a message along with the number of times it was added.

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

type: ReactionTypeUnion

Type of the reaction

total_count: int

Number of times the reaction was added

ReactionType

class aiogram.types.reaction_type.ReactionType(**extra_data: Any)

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

- aiogram.types.reaction_type_emoji.ReactionTypeEmoji
- aiogram.types.reaction_type_custom_emoji.ReactionTypeCustomEmoji
- aiogram.types.reaction_type_paid.ReactionTypePaid

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

ReactionTypeCustomEmoji

class aiogram.types.reaction_type_custom_emoji.ReactionTypeCustomEmoji(*, type: Lit-

eral[ReactionTypeType.CUSTOM_EMOJI]
= ReactionTypeType.CUSTOM_EMOJI,
custom_emoji_id: str,
**extra_data: Any)

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'

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'

emoji: str

ReactionTypePaid

The reaction is paid.

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

type: Literal[ReactionTypeType.PAID]

Type of the reaction, always 'paid'

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.

one_time_keyboard: bool | None

Optional. Requests clients to hide the keyboard as soon as it's been used. The keyboard will still be available, but clients will automatically display the usual letter-keyboard in the chat - the user can press a special button in the input field to see the custom keyboard again. Defaults to *false*.

input_field_placeholder: str | None

Optional. The placeholder to be shown in the input field when the keyboard is active; 1-64 characters

selective: bool | None

Optional. Use this parameter if you want to show the keyboard to specific users only. Targets: 1) users that are @mentioned in the *text* of the *aiogram.types.message.Message* object; 2) if the bot's message is a reply to a message in the same chat and forum topic, sender of the original message.

ReplyKeyboardRemove

Upon receiving a message with this object, Telegram clients will remove the current custom keyboard and display the default letter-keyboard. By default, custom keyboards are displayed until a new keyboard is sent by a bot. An exception is made for one-time keyboards that are hidden immediately after the user presses a button (see <code>aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup</code>). Not supported in channels and for messages sent on behalf of a Telegram Business account.

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

remove_keyboard: Literal[True]

Requests clients to remove the custom keyboard (user will not be able to summon this keyboard; if you want to hide the keyboard from sight but keep it accessible, use *one_time_keyboard* in *aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup*)

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

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: ChatIdUnion | 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.

```
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.

```
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

**extra data: Any)

```
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
     id: int
          Unique identifier for the story in the chat
SwitchInlineQueryChosenChat
class aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQueryChosenChat(*, query: str
                                                                                                | None =
                                                                                                None, al-
                                                                                                low_user_chats:
                                                                                                bool | None
                                                                                                = None, al-
                                                                                                low_bot_chats:
                                                                                                bool | None
                                                                                                = None, al-
                                                                                                low_group_chats:
                                                                                                bool | None
                                                                                                = None, al-
                                                                                                low_channel_chats:
                                                                                                bool | None
                                                                                                = None,
```

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
```

```
allow_group_chats: bool | None
```

Optional. True, if group and supergroup chats can be chosen

```
allow_channel_chats: bool | None
```

Optional. True, if channel chats can be chosen

TextQuote

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

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

text: str

Text of the quoted part of a message that is replied to by the given message

position: int

Approximate quote position in the original message in UTF-16 code units as specified by the sender

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
language_code: str | None
```

Optional. User's or bot's username

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.

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.

has_main_web_app: bool | None

Optional. True, if the bot has a main Web App. Returned only in aiogram.methods.get_me.GetMe.

property full_name:

property url: str

mention_markdown($name: str \mid None = None$) $\rightarrow str$

mention_html ($name: str \mid None = None$) $\rightarrow str$

get_profile_photos(offset: int | None = None, limit: int | None = None, **kwargs: Any) \rightarrow GetUserProfilePhotos

Shortcut for method aiogram.methods.get_user_profile_photos.GetUserProfilePhotos will automatically fill method attributes:

• user_id

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

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

instance method aiogram.methods.get_user_profile_photos. of GetUserProfilePhotos

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

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

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

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.

user_ids: list[int] | None

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

Deprecated since version API:7.2: https://core.telegram.org/bots/api-changelog#march-31-2024

Venue

```
class aiogram.types.venue.Venue(*, location: Location, title: str, address: str, foursquare_id: str | None = None, foursquare_type: str | None = None, google_place_id: str | None = None, google_place_type: str | None = None, **extra_data: Any)
```

This object represents a venue.

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

location: Location

Venue location. Can't be a live location

title: str

Name of the venue

address: str

Address of the venue

foursquare_id: str | None

Optional. Foursquare identifier of the venue

foursquare_type: str | None

Optional. Foursquare type of the venue. (For example, 'arts_entertainment/default', 'arts_entertainment/aquarium' or 'food/icecream'.)

google_place_id: str | None

Optional. Google Places identifier of the venue

google_place_type: str | None

Optional. Google Places type of the venue. (See supported types.)

Video

This object represents a video file.

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

file_id: str

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

file_unique_id: str

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

width: int

Video width as defined by the sender

height: int

Video height as defined by the sender

duration: int

Duration of the video in seconds as defined by the sender

thumbnail: PhotoSize | None

Optional. Video thumbnail

cover: list[PhotoSize] | None

Optional. Available sizes of the cover of the video in the message

start_timestamp: datetime.datetime | None

Optional. Timestamp in seconds from which the video will play in the message

file_name: str | None

Optional. Original filename as defined by the sender

mime_type: str | None

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

file_size: int | None

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

VideoChatEnded

```
class aiogram.types.video_chat_ended.VideoChatEnded(*, duration: int, **extra_data: Any)
```

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

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

duration: int

Video chat duration in seconds

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

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

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

VideoNote

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

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

file_id: str

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

file_unique_id: str

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

length: int

Video width and height (diameter of the video message) as defined by the sender

duration: int

Duration of the video in seconds as defined by the sender

thumbnail: *PhotoSize* | None

Optional. Video thumbnail

file_size: int | None

Optional. File size in bytes

Voice

This object represents a voice note.

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

file id: str

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

file_unique_id: str

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

duration: int

Duration of the audio in seconds as defined by the sender

mime_type: str | None

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

file_size: int | None

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

WebAppData

class aiogram.types.web_app_data.WebAppData(*, data: str, button_text: str, **extra_data: Any)

Describes data sent from a Web App to the bot.

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

data: str

The data. Be aware that a bad client can send arbitrary data in this field.

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

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

```
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

```
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

```
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[InlineQueryResultUnion], 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) → AnswerInlineQuery

Shortcut for method aiogram.methods.answer_inline_query.AnswerInlineQuery will automatically fill method attributes:

• inline_query_id

Use this method to send answers to an inline query. On success, True is returned.

No more than 50 results per query are allowed.

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

Parameters

- **results** A JSON-serialized array of results for the inline query
- **cache_time** The maximum amount of time in seconds that the result of the inline query may be cached on the server. Defaults to 300.
- **is_personal** Pass True if results may be cached on the server side only for the user that sent the query. By default, results may be returned to any user who sends the same query.
- **next_offset** Pass the offset that a client should send in the next query with the same text to receive more results. Pass an empty string if there are no more results or if you don't support pagination. Offset length can't exceed 64 bytes.
- button A JSON-serialized object describing a button to be shown above inline query results
- **switch_pm_parameter** Deep-linking parameter for the /start message sent to the bot when user presses the switch button. 1-64 characters, only A-Z, a-z, 0-9, _ and are allowed
- **switch_pm_text** If passed, clients will display a button with specified text that switches the user to a private chat with the bot and sends the bot a start message with the parameter *switch_pm_parameter*

Returns

instance of method aiogram.methods.answer_inline_query.AnswerInlineQuery

InlineQueryResult

class aiogram.types.inline_query_result.InlineQueryResult(**extra_data: Any)

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

- aiogram.types.inline_query_result_cached_audio.InlineQueryResultCachedAudio
- aiogram.types.inline_query_result_cached_document.InlineQueryResultCachedDocument
- aiogram.types.inline_query_result_cached_gif.InlineQueryResultCachedGif
- aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResultCachedMpeg4Gif
- aiogram.types.inline_query_result_cached_photo.InlineQueryResultCachedPhoto

- aiogram.types.inline_query_result_cached_sticker.InlineQueryResultCachedSticker
- aiogram.types.inline_query_result_cached_video.InlineQueryResultCachedVideo
- aiogram.types.inline_query_result_cached_voice.InlineQueryResultCachedVoice
- aiogram.types.inline_query_result_article.InlineQueryResultArticle
- aiogram.types.inline_query_result_audio.InlineQueryResultAudio
- aiogram.types.inline_query_result_contact.InlineQueryResultContact
- $\bullet \ aiogram.types.inline_query_result_game.InlineQueryResultGame\\$
- 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

InlineQueryResultArticle

class aiogram.types.inline_query_result_article.InlineQueryResultArticle(*, type: Lit-

eral[InlineQueryResultType.ARTICLE] = InlineQueryResult-Type.ARTICLE, id: str, title: str, input_message_content: InputTextMessage-Content | InputLocationMessageContent | InputVenueMessage-Content | InputContactMessageContent | InputInvoiceMessageContent, reply_markup: InlineKeyboardMarkup | None = None, url: $str \mid None = None,$ description: str | None = None,thumbnail_url: str | None = None. thumbnail width: int | None = None,thumbnail_height: $int \mid None = None,$ hide_url: bool | 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: InputMessageContentUnion

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

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

hide_url: bool | None

Optional. Pass True if you don't want the URL to be shown in the message

Deprecated since version API:8.2: https://core.telegram.org/bots/api-changelog#january-1-2025

InlineQueryResultAudio

```
class aiogram.types.inline_query_result_audio.InlineQueryResultAudio(*, type: ~typ-
```

```
ing.Literal[InlineQueryResultType.AUDIO]
= Inline Query Result-
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:
list[~aiogram.types.message_entity.Message_
| None = None, performer:
str \mid None = None,
audio_duration: int | None
= None, reply\_markup:
~aiogram.types.inline_keyboard_markup.Inl
| None = None,
input message content:
~aiogram.types.input_text_message_content.
~aiogram.types.input_location_message_com
~aiogram.types.input_venue_message_conten
~aiogram.types.input_contact_message_cont
~aiogram.types.input_invoice_message_cont
| None = None,
**extra_data:
~typing.Any)
```

Represents a link to an MP3 audio file. 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#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.
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: InputMessageContentUnion | None
     Optional. Content of the message to be sent instead of the audio
```

In line Query Result Cached Audio

```
\textbf{class} \  \, \textbf{aiogram.types.inline\_query\_result\_cached\_audio.InlineQueryResultCachedAudio}(*, \textit{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:
list[~aiogram.types.message
|None|
None, re-
ply_markup:
~aiogram.types.inline_keyb
| None =
None, in-
put_message_content:
~aiogram.types.input_text_i
~aiogram.types.input_locat
~aiogram.types.input_venue
~aiogram.types.input_conta
~aiogram.types.input_invoid
| None =
None,
**ex-
tra_data:
~typ-
ing.Any)
```

Represents a link to 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.AUDI0]

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

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

 ${\tt reply_markup:} \quad \textit{InlineKeyboardMarkup} \; \mid \; {\tt None}$

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | 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:
                                                                                                           list[~aiogram.types.
                                                                                                           None
                                                                                                           None,
                                                                                                           re-
                                                                                                           ply_markup:
                                                                                                           ~aiogram.types.inlin
                                                                                                           None
```

Chapter 2. Contentsone,

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

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: InputMessageContentUnion | None

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

InlineQueryResultCachedGif

```
\textbf{class} \  \, \textbf{aiogram.types.inline\_query\_result\_cached\_gif.} \\ \textbf{InlineQueryResultCachedGif}(*, \textit{type: } \textit{~type: } 
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    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:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    list[~aiogram.types.message_ent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |None = None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    show_caption_above_media:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    bool |
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     ~aiogram.client.default.Default
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |None = < De-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   fault('show_caption_above_medi
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    reply_markup:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    ~aiogram.types.inline_keyboard_
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | None =
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    None, in-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    put_message_content:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     ~aiogram.types.input_text_messc
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     ~aiogram.types.input_location_n
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     ~aiogram.types.input_venue_mes
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    ~aiogram.types.input_contact_m
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    ~aiogram.types.input_invoice_m
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | None =
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     **extra_data:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    ~typing.Any)
```

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

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

type: Literal[InlineQueryResultType.GIF]

Type of the result, must be gif

id: str

```
Unique identifier for this result, 1-64 bytes
gif_file_id: str
     A valid file identifier for the GIF file
title: str | None
     Optional. Title for the result
caption: str | None
     Optional. Caption of the GIF file to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the caption. See formatting options for more details.
caption_entities: list[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputMessageContentUnion | None
```

Optional. Content of the message to be sent instead of the GIF animation

InlineQueryResultCachedMpeg4Gif

290

```
class aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResultCachedMpeg4Gif(*,
                                                                                                          type:
                                                                                                          ing.Literal[InlineQ
                                                                                                          In-
                                                                                                          line-
                                                                                                          QueryRe-
                                                                                                          sult-
                                                                                                          Type.MPEG4_GIF
                                                                                                          str,
                                                                                                          mpeg4_file_id:
                                                                                                          str,
                                                                                                          ti-
                                                                                                          tle:
                                                                                                          str
                                                                                                          None
                                                                                                          None,
                                                                                                          cap-
                                                                                                          tion:
                                                                                                          str
                                                                                                          None
                                                                                                          None,
                                                                                                          parse_mode:
                                                                                                          str
                                                                                                          ~aiogram.client.de
                                                                                                          None
                                                                                                          <De-
                                                                                                          fault('parse_mode'
                                                                                                          cap-
                                                                                                          tion\_entities:
                                                                                                          list[~aiogram.type
                                                                                                          None
                                                                                                          None,
                                                                                                          show_caption_abo
                                                                                                          bool
                                                                                                           ~aiogram.client.de
                                                                                                          None
                                                                                                           <De-
                                                                                                          fault('show_captio
                                                                                                          ply_markup:
```

Chapter 2. Contents

~aiogram.types.ini

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

Source: https://core.telegram.org/bots/api#inlinequeryresultcachedmpeg4gif

type: Literal[InlineQueryResultType.MPEG4_GIF]

Type of the result, must be mpeg4_gif

id: str

Unique identifier for this result, 1-64 bytes

mpeg4_file_id: str

A valid file identifier for the MPEG4 file

title: str | None

Optional. Title for the result

caption: str | None

Optional. Caption of the MPEG-4 file to be sent, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Optional. Mode for parsing entities in the caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | None

Optional. Content of the message to be sent instead of the video animation

InlineQueryResultCachedPhoto

```
class aiogram.types.inline_query_result_cached_photo.InlineQueryResultCachedPhoto(*, type:
                                                                                                     ~typ-
                                                                                                     ing.Literal[InlineQueryRest
                                                                                                     = In line-
                                                                                                     QueryRe-
                                                                                                     sult-
                                                                                                     Type.PHOTO,
                                                                                                     id: str,
                                                                                                     photo_file_id:
                                                                                                     str, title:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     descrip-
                                                                                                     tion: str |
                                                                                                     None =
                                                                                                     None,
                                                                                                     caption:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     parse_mode:
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('parse_mode')>,
                                                                                                     cap-
                                                                                                     tion_entities:
                                                                                                     list[~aiogram.types.message
                                                                                                     | None =
                                                                                                     None,
                                                                                                     show_caption_above_media
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('show_caption_above_
                                                                                                     re-
                                                                                                     ply_markup:
                                                                                                     ~aiogram.types.inline_keyb
                                                                                                     | None =
                                                                                                     None, in-
                                                                                                     put_message_content:
                                                                                                     ~aiogram.types.input_text_i
                                                                                                     ~aiogram.types.input_locat
                                                                                                     ~aiogram.types.input_venue
                                                                                                     ~aiogram.types.input_conta
                                                                                                     ~aiogram.types.input_invoid
                                                                                                     | None =
                                                                                                     None.
                                                                                                     **ex-
                                                                                                     tra_data:
                                                                                                      ~typ-
```

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

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

type: Literal[InlineQueryResultType.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

caption: str | None

Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Optional. Mode for parsing entities in the photo caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | None

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

InlineQueryResultCachedSticker

```
{\bf class} \ {\bf aiogram.types.inline\_query\_result\_cached\_sticker.} {\bf InlineQueryResultCachedSticker} (*, {\bf class}) {\bf class} \ {\bf class
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             type:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             eral[InlineQueryResul
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             line-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              QueryRe-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sult-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Type.STICKER,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             id:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sticker_file_id:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             str,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              re-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ply_markup:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             lineKey-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             board-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Markup
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             None
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             None,
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             in-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put_message_content:
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             TextMes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              Lo-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             ca-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tion-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Mes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             putV-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             enueMes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             put-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tactMes-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             sage-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             Con-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             tent
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             In-
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              putIn-
```

2.3. Bot API

295eMes-

sage-Con-

tent

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

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

type: Literal[InlineQueryResultType.STICKER]

Type of the result, must be sticker

id: str

Unique identifier for this result, 1-64 bytes

sticker_file_id: str

A valid file identifier of the sticker

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | None

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

InlineQueryResultCachedVideo

~typ-ing.Any)

```
class aiogram.types.inline_query_result_cached_video.InlineQueryResultCachedVideo(*, type:
                                                                                                     ~typ-
                                                                                                     ing.Literal[InlineQueryRest
                                                                                                     = Inline-
                                                                                                     QueryRe-
                                                                                                     sult-
                                                                                                     Type.VIDEO,
                                                                                                     id: str,
                                                                                                     video_file_id:
                                                                                                     str, title:
                                                                                                     str, de-
                                                                                                     scription:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     caption:
                                                                                                     str | None
                                                                                                     = None,
                                                                                                     parse_mode:
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('parse_mode')>,
                                                                                                     cap-
                                                                                                     tion_entities:
                                                                                                     list[~aiogram.types.message
                                                                                                     | None =
                                                                                                     None,
                                                                                                     show_caption_above_media
                                                                                                     ~aiogram.client.default.Def
                                                                                                     | None =
                                                                                                     <De-
                                                                                                     fault('show_caption_above_
                                                                                                     ply_markup:
                                                                                                     ~aiogram.types.inline_keyb
                                                                                                     | None =
                                                                                                     None, in-
                                                                                                     put_message_content:
                                                                                                     ~aiogram.types.input_text_i
                                                                                                     ~aiogram.types.input_locat
                                                                                                     ~aiogram.types.input_venue
                                                                                                     ~aiogram.types.input_conta
                                                                                                     ~aiogram.types.input_invoid
                                                                                                     | None =
                                                                                                     None,
                                                                                                     **ex-
                                                                                                     tra_data:
```

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

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

type: Literal[InlineQueryResultType.VIDE0]

Type of the result, must be video

id: str

Unique identifier for this result, 1-64 bytes

video_file_id: str

A valid file identifier for the video file

title: str

Title for the result

description: str | None

Optional. Short description of the result

caption: str | None

Optional. Caption of the video to be sent, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Optional. Mode for parsing entities in the video caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | None

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

InlineQueryResultCachedVoice

```
class aiogram.types.inline_query_result_cached_voice.InlineQueryResultCachedVoice(*, type:
```

```
~typ-
ing.Literal[InlineQueryRest
= Inline-
QueryRe-
sult-
Type.VOICE,
id: str,
voice_file_id:
str, title:
str.
caption:
str | None
= None,
parse_mode:
str |
~aiogram.client.default.Def
|None|
<De-
fault('parse_mode')>,
cap-
tion_entities:
list[~aiogram.types.message
|None|
None, re-
ply_markup:
~aiogram.types.inline_keyb
| None =
None, in-
put_message_content:
~aiogram.types.input_text_i
~aiogram.types.input_locat
~aiogram.types.input_venue
~aiogram.types.input_conta
~aiogram.types.input_invoid
| None =
None,
**ex-
tra_data:
~typ-
ing.Any)
```

Represents a link to a 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

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: InputMessageContentUnion | None

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

InlineQueryResultContact

class aiogram.types.inline_query_result_contact.InlineQueryResultContact(*, type: Lit-

eral[InlineQueryResultType.CONTACT

= InlineQueryResult-

Type.CONTACT, *id*:

str, phone_number:

str, first_name: str,

last_name: str | None

= None, vcard: str

None = None,

reply_markup: In-

lineKeyboardMarkup

| None = None, in-

put_message_content:

InputTextMessage-

Content |

InputLocationMes-

sageContent

InputVenueMessage-

Content

InputContactMes-

sageContent |

InputInvoiceMes-

sageContent | 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

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | 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:
list[~aiogram.types.message_entity.M
| None = None,
description: str |
None = None,
reply_markup:
~aiogram.types.inline_keyboard_max
| None = None, in-
put_message_content:
~aiogram.types.input_text_message_
~aiogram.types.input_location_mess
~aiogram.types.input_venue_message
~aiogram.types.input_contact_messa
~aiogram.types.input_invoice_messa
| None = None,
thumbnail_url: str
| None = None,
thumbnail_width:
int \mid None = None,
thumbnail_height:
int \mid None = None,
**extra\_data:
~typing.Any)
```

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

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

type: Literal[InlineQueryResultType.DOCUMENT]

Type of the result, must be document

id: str

Unique identifier for this result, 1-64 bytes

title: str

Title for the result

```
document_url: str
          A valid URL for the file
     mime_type: str
          MIME type of the content of the file, either 'application/pdf' or 'application/zip'
     caption: str | None
          Optional. Caption of the document to be sent, 0-1024 characters after entities parsing
     parse_mode: str | Default | None
          Optional. Mode for parsing entities in the document caption. See formatting options for more details.
     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: InputMessageContentUnion | None
          Optional. Content of the message to be sent instead of the file
     thumbnail_url: str | None
          Optional. URL of the thumbnail (JPEG only) for the file
     thumbnail_width: int | None
          Optional. Thumbnail width
     thumbnail_height: int | None
          Optional. Thumbnail height
InlineQueryResultGame
class aiogram.types.inline_query_result_game.InlineQueryResultGame(*, type: Lit-
                                                                              eral[InlineQueryResultType.GAME]
                                                                               = InlineQueryResult-
                                                                               Type.GAME, id: str,
                                                                               game_short_name: str,
                                                                               reply markup:
                                                                               InlineKeyboardMarkup |
                                                                               None = None, **extra data:
                                                                              Any)
     Represents a Game.
     Source: https://core.telegram.org/bots/api#inlinequeryresultgame
     type: Literal[InlineQueryResultType.GAME]
          Type of the result, must be game
     id: str
          Unique identifier for this result, 1-64 bytes
     game_short_name: str
          Short name of the game
```

reply_markup: InlineKeyboardMarkup | None Optional. Inline keyboard attached to the message

InlineQueryResultGif

```
class aiogram.types.inline_query_result_gif.InlineQueryResultGif(*, type: ~typ-
```

```
ing.Literal[InlineQueryResultType.GIF]
= InlineQueryResultType.GIF,
id: str, gif_url: str,
thumbnail_url: str, gif_width:
int | None = None, gif_height:
int \mid None = None,
gif\_duration: int \mid None =
None, thumbnail_mime_type:
str \mid None = None, title: str \mid
None = None, caption: str \mid
None = None, parse\_mode: str
~aiogram.client.default.Default
|None|
<Default('parse_mode')>,
caption_entities:
list[~aiogram.types.message_entity.MessageEntity
| None = None,
show_caption_above_media:
bool
~aiogram.client.default.Default
|None = < De-
fault('show_caption_above_media')>,
reply_markup:
~aiogram.types.inline_keyboard_markup.InlineKe
| None = None,
input_message_content:
~aiogram.types.input_text_message_content.Input
~aiogram.types.input_location_message_content.l
~aiogram.types.input_venue_message_content.Inp
~aiogram.types.input_contact_message_content.In
~aiogram.types.input_invoice_message_content.Ir
| None = None, **extra_data:
~typing.Any)
```

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

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

type: Literal[InlineQueryResultType.GIF]

Type of the result, must be gif

```
id: str
     Unique identifier for this result, 1-64 bytes
gif_url: str
     A valid URL for the GIF file
thumbnail_url: str
     URL of the static (JPEG or GIF) or animated (MPEG4) thumbnail for the result
gif_width: int | None
     Optional. Width of the GIF
gif_height: int | None
     Optional. Height of the GIF
gif_duration: int | None
     Optional. Duration of the GIF in seconds
thumbnail_mime_type: str | None
     Optional. MIME type of the thumbnail, must be one of 'image/jpeg', 'image/gif', or 'video/mp4'. Defaults
     to 'image/jpeg'
title: str | None
     Optional. Title for the result
caption: str | None
     Optional. Caption of the GIF file to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the caption. See formatting options for more details.
caption_entities: list[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputMessageContentUnion | None
```

InlineQueryResultLocation

2.3. Bot API 305

Optional. Content of the message to be sent instead of the GIF animation

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 | None = None,
thumbnail_height:
int | None = None,
**extra data:

Any)

Represents a location on a map. By default, the location will be sent by the user. Alternatively, you can use <code>input_message_content</code> 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.

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: InputMessageContentUnion | None

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

thumbnail_url: str | None

Optional. Url of the thumbnail for the result

thumbnail_width: int | None

Optional. Thumbnail width

thumbnail_height: int | None

Optional. Thumbnail height

InlineQueryResultMpeg4Gif

```
class aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMpeg4Gif(*, type: ~typ-
                                                                                               ing.Literal[InlineQueryResultType.I
                                                                                               = Inline-
                                                                                               QueryResult-
                                                                                               Type.MPEG4_GIF,
                                                                                               id: str,
                                                                                               mpeg4_url: str,
                                                                                               thumbnail_url:
                                                                                               str,
                                                                                               mpeg4_width:
                                                                                               int \mid None =
                                                                                               None,
                                                                                               mpeg4_height:
                                                                                               int | None =
                                                                                               None,
                                                                                               mpeg4_duration:
                                                                                               int | None =
                                                                                               None, thumb-
                                                                                               nail_mime_type:
                                                                                               str \mid None =
                                                                                               None, title: str |
                                                                                               None = None,
                                                                                               caption: str |
                                                                                               None = None,
                                                                                               parse_mode: str |
                                                                                               ~aiogram.client.default.Default
                                                                                               |None = < De-
                                                                                               fault('parse_mode')>,
                                                                                               caption_entities:
                                                                                               list[~aiogram.types.message_entity.
                                                                                               | None = None,
                                                                                               show_caption_above_media:
                                                                                               bool |
                                                                                               \hbox{$\sim$}aiogram.client.default.Default
                                                                                               |None = < De-
                                                                                               fault('show_caption_above_media')
                                                                                               reply_markup:
                                                                                               ~aiogram.types.inline_keyboard_m
                                                                                               | None = None,
                                                                                               in-
                                                                                               put_message_content:
                                                                                               ~aiogram.types.input_text_message
                                                                                               ~aiogram.types.input_location_mes
                                                                                               ~aiogram.types.input_venue_messa
                                                                                               ~aiogram.types.input_contact_mess
                                                                                               ~aiogram.types.input_invoice_mess
                                                                                               | None = None,
                                                                                               **extra_data:
                                                                                               ~typing.Any)
```

Represents a link to a video animation (H.264/MPEG-4 AVC video without sound). By default, this animated

MPEG-4 file will be sent by the user with optional caption. Alternatively, you can use *input_message_content* 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

thumbnail_url: str

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

mpeg4_width: int | None

Optional. Video width

mpeg4_height: int | None

Optional. Video height

mpeg4_duration: int | None

Optional. Video duration in seconds

thumbnail_mime_type: str | None

Optional. MIME type of the thumbnail, must be one of 'image/jpeg', 'image/gif', or 'video/mp4'. Defaults to 'image/jpeg'

title: str | None

Optional. Title for the result

caption: str | None

Optional. Caption of the MPEG-4 file to be sent, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Optional. Mode for parsing entities in the caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Optional. Pass True, if the caption must be shown above the message media

reply_markup: InlineKeyboardMarkup | None

Optional. Inline keyboard attached to the message

input_message_content: InputMessageContentUnion | 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:
                                                                                     list[~aiogram.types.message_entity.Message_
                                                                                     | None = None,
                                                                                     show_caption_above_media:
                                                                                     bool |
                                                                                     ~aiogram.client.default.Default
                                                                                     |None = < De-
                                                                                    fault('show_caption_above_media')>,
                                                                                     reply_markup:
                                                                                     ~aiogram.types.inline_keyboard_markup.Inl
                                                                                     | None = None,
                                                                                     input_message_content:
                                                                                     ~aiogram.types.input_text_message_content.
                                                                                     ~aiogram.types.input_location_message_com
                                                                                     ~aiogram.types.input_venue_message_conten
                                                                                     ~aiogram.types.input_contact_message_cont
                                                                                     ~aiogram.types.input_invoice_message_cont
                                                                                     | None = None,
                                                                                     **extra_data:
                                                                                     ~typing.Any)
```

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

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

```
type: Literal[InlineQueryResultType.PHOTO]
```

Type of the result, must be photo

id: str

Unique identifier for this result, 1-64 bytes

photo_url: str

A valid URL of the photo. Photo must be in JPEG format. Photo size must not exceed 5MB

```
thumbnail_url: str
     URL of the thumbnail for the photo
photo_width: int | None
     Optional. Width of the photo
photo_height: int | None
     Optional. Height of the photo
title: str | None
     Optional. Title for the result
description: str | None
     Optional. Short description of the result
caption: str | None
     Optional. Caption of the photo to be sent, 0-1024 characters after entities parsing
parse_mode: str | Default | None
     Optional. Mode for parsing entities in the photo caption. See formatting options for more details.
caption_entities: list[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputMessageContentUnion | 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] = InlineQueryResult-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 | InputVenueMessageContent 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'.)
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: InputMessageContentUnion | 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
```

In line Query Result Video

| None = None, **extra_data: ~typing.Any)

```
class aiogram.types.inline_query_result_video.InlineQueryResultVideo(*, type: ~typ-
                                                                                     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:
                                                                                     list[~aiogram.types.message_entity.Message_
                                                                                     | None = None,
                                                                                     show_caption_above_media:
                                                                                     bool |
                                                                                     ~aiogram.client.default.Default
                                                                                     |None = < De-
                                                                                    fault('show_caption_above_media')>,
                                                                                     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
                                                                                     ~aiogram.types.input_contact_message_cont
                                                                                     ~aiogram.types.input_invoice_message_cont
```

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

If an InlineQueryResultVideo message contains an embedded video (e.g., YouTube), you **must** replace its content using <code>input_message_content</code>.

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

type: Literal[InlineQueryResultType.VIDE0]

Type of the result, must be video

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.
caption_entities: list[MessageEntity] | None
     Optional. List of special entities that appear in the caption, which can be specified instead of parse_mode
show_caption_above_media: bool | Default | None
     Optional. Pass True, if the caption must be shown above the message media
video_width: int | None
     Optional. Video width
video_height: int | None
     Optional. Video height
video_duration: int | None
     Optional. Video duration in seconds
description: str | None
     Optional. Short description of the result
reply_markup: InlineKeyboardMarkup | None
     Optional. Inline keyboard attached to the message
input_message_content: InputMessageContentUnion | None
```

Optional. Content of the message to be sent instead of the video. This field is required if InlineQueryRe-

sultVideo is used to send an HTML-page as a result (e.g., a YouTube video).

InlineQueryResultVoice

```
class aiogram.types.inline_query_result_voice.InlineQueryResultVoice(*, type: ~type-
                                                                                      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:
                                                                                      list[~aiogram.types.message_entity.Message_
                                                                                      | None = None,
                                                                                      voice_duration: int | None
                                                                                      = None, reply\_markup:
                                                                                      ~aiogram.types.inline_keyboard_markup.Inl
                                                                                      | None = None,
                                                                                      input message content:
                                                                                      ~aiogram.types.input_text_message_content.
                                                                                      ~aiogram.types.input_location_message_com
                                                                                      ~aiogram.types.input_venue_message_conten
                                                                                      ~aiogram.types.input_contact_message_cont
                                                                                      ~aiogram.types.input_invoice_message_cont
                                                                                      | None = None,
                                                                                      **extra_data:
                                                                                      ~typing.Any)
     Represents a link to a voice recording in an .OGG container encoded with OPUS. By default, this voice recording
     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

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: InputMessageContentUnion | 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{class} \  \, \textbf{aiogram.types.inline\_query\_results\_button.} \\ \textbf{InlineQueryResultsButton} (*, \textit{text: str}, \textit{web\_app: } \\ \textbf{class} \  \, \textbf{aiogram.types.inline\_query\_results\_button.} \\ \textbf{discourse} (*, \textit{text: str}, \textit{web\_app: } \\ \textbf{discourse} (*, \textit{text: str}, \textit{text:
```

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.

 $\textbf{Source:}\ https://core.telegram.org/bots/api\#inline query results button$

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.

```
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

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 3.19.0
class aiogram.types.input_invoice_message_content.InputInvoiceMessageContent(*, title: str,
                                                                                              description: str,
                                                                                              payload: str,
                                                                                              currency: str,
                                                                                              prices:
                                                                                              list[LabeledPrice],
                                                                                              provider_token:
                                                                                              str \mid None =
                                                                                              None,
                                                                                              max_tip_amount:
                                                                                              int \mid None =
                                                                                              None, sug-
                                                                                              gested_tip_amounts:
                                                                                              list[int] | None
                                                                                              = None,
                                                                                              provider_data:
                                                                                              str | None =
                                                                                              None,
                                                                                              photo_url: str |
                                                                                              None = None,
                                                                                              photo_size: int |
                                                                                              None = None,
                                                                                              photo_width:
                                                                                              int | None =
                                                                                              None,
                                                                                              photo_height:
                                                                                              int \mid None =
                                                                                              None,
                                                                                              need_name:
                                                                                              bool | None =
                                                                                              None,
                                                                                              need_phone_number:
                                                                                              bool | None =
                                                                                              None,
                                                                                              need email:
                                                                                              bool \mid None =
                                                                                              None,
                                                                                              need_shipping_address:
                                                                                              bool | 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

Any)

is_flexible: bool
| None = None,
**extra_data:

Product name, 1-32 characters

description: str

Product description, 1-255 characters

payload: str

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

currency: str

Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

prices: list[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

provider_token: str | None

Optional. Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

max_tip_amount: int | None

Optional. The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.

suggested_tip_amounts: list[int] | None

Optional. A JSON-serialized array of suggested amounts of tip in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.

provider_data: str | None

Optional. A JSON-serialized object for data about the invoice, which will be shared with the payment provider. A detailed description of the required fields should be provided by the payment provider.

photo_url: str | None

Optional. URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service.

photo_size: int | None

Optional. Photo size in bytes

photo_width: int | None

Optional. Photo width

photo_height: int | None

Optional. Photo height

need_name: bool | None

Optional. Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.

need_phone_number: bool | None

Optional. Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.

need_email: bool | None

Optional. Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.

need_shipping_address: bool | None

Optional. Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

send_phone_number_to_provider: bool | None

Optional. Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.

send_email_to_provider: bool | None

Optional. Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.

is_flexible: bool | None

Optional. Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

InputLocationMessageContent

class aiogram.types.input_location_message_content.InputLocationMessageContent(*, latitude:

float, longitude: float, horizontal_accuracy: float | None = None,*live_period: int* | *None* = None, heading: int | None = None, proximity alert radius: *int* | *None* = None, **ex-

tra_data:
Any)

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

bool | None = None,
**extra_data:
~typing.Any)

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

InputTextMessageContent

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
     link_preview_options: LinkPreviewOptions | Default | None
          Optional. Link preview generation options for the message
     disable_web_page_preview: bool | None
          Optional. Disables link previews for links in the sent message
          Deprecated since version API:7.0: https://core.telegram.org/bots/api-changelog#december-29-2023
InputVenueMessageContent
class aiogram.types.input_venue_message_content.InputVenueMessageContent(*, latitude: float,
                                                                                       longitude: float, title:
                                                                                       str, address: str,
                                                                                       foursquare id: str |
                                                                                       None = None,
                                                                                       foursquare_type: str
                                                                                       | None = None,
                                                                                       google_place_id: str
                                                                                       | None = None,
                                                                                       google_place_type:
                                                                                       str \mid 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
     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.)

PreparedInlineMessage

Describes an inline message to be sent by a user of a Mini App.

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

id: str

Unique identifier of the prepared message

expiration_date: DateTimeUnion

Expiration date of the prepared message, in Unix time. Expired prepared messages can no longer be used

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.

Payments

AffiliateInfo

Contains information about the affiliate that received a commission via this transaction.

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

```
commission_per_mille: int
```

The number of Telegram Stars received by the affiliate for each 1000 Telegram Stars received by the bot from referred users

amount: int

Integer amount of Telegram Stars received by the affiliate from the transaction, rounded to 0; can be negative for refunds

affiliate_user: *User* | None

Optional. The bot or the user that received an affiliate commission if it was received by a bot or a user

affiliate_chat: Chat | None

Optional. The chat that received an affiliate commission if it was received by a chat

nanostar_amount: int | None

Optional. The number of 1/1000000000 shares of Telegram Stars received by the affiliate; from -999999999 to 999999999; can be negative for refunds

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

currency: str

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars

total_amount: int

Total price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45 pass amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

LabeledPrice

class aiogram.types.labeled_price.LabeledPrice(*, label: str, amount: int, **extra_data: Any)

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

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

label: str

Portion label

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

name: str | None
Optional. User name

phone_number: str | None
Optional. User's phone number

email: str | None
Optional. User email

shipping_address: ShippingAddress | None

Optional. User shipping address

PaidMediaPurchased

This object contains information about a paid media purchase.

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

from_user: User

User who purchased the media

paid_media_payload: str

Bot-specified paid media payload

PreCheckoutQuery

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

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

id: str

Unique query identifier

from_user: User

User who sent the query

currency: str

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars

total_amount: int

Total price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45 pass amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

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

• pre_checkout_query_id

Once the user has confirmed their payment and shipping details, the Bot API sends the final confirmation in the form of an *aiogram.types.update.Update* with the field *pre_checkout_query*. Use this method to respond to such pre-checkout queries. On success, True is returned. **Note:** The Bot API must receive an answer within 10 seconds after the pre-checkout query was sent.

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

Parameters

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

Returns

```
instance of method aiogram.methods.answer_pre_checkout_query.
AnswerPreCheckoutQuery
```

RefundedPayment

This object contains basic information about a refunded payment.

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

```
currency: Literal['XTR']
```

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars. Currently, always 'XTR'

total_amount: int

Total refunded price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45, total_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

```
invoice_payload: str
```

Bot-specified invoice payload

telegram_payment_charge_id: str

Telegram payment identifier

```
provider_payment_charge_id: str | None
```

Optional. Provider payment identifier

RevenueWithdrawalState

class aiogram.types.revenue_withdrawal_state.RevenueWithdrawalState(**extra_data: Any)

This object describes the state of a revenue withdrawal operation. Currently, it can be one of

- aiogram.types.revenue_withdrawal_state_pending.RevenueWithdrawalStatePending
- aiogram.types.revenue_withdrawal_state_succeeded.RevenueWithdrawalStateSucceeded
- aiogram.types.revenue_withdrawal_state_failed.RevenueWithdrawalStateFailed

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

RevenueWithdrawalStateFailed

class aiogram.types.revenue_withdrawal_state_failed.RevenueWithdrawalStateFailed(*, type:

eral[RevenueWithdrawalState = RevenueWithdrawal-

State-Type.FAILED,

**extra_data: Any)

The withdrawal failed and the transaction was refunded.

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

type: Literal[RevenueWithdrawalStateType.FAILED]

Type of the state, always 'failed'

RevenueWithdrawalStatePending

The withdrawal is in progress.

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

type: Literal[RevenueWithdrawalStateType.PENDING]

Type of the state, always 'pending'

RevenueWithdrawalStateSucceeded

```
class aiogram.types.revenue_withdrawal_state_succeeded.RevenueWithdrawalStateSucceeded(*,
```

```
type:
Lit-
eral[RevenueWithdra
Rev-
enue-
With-
drawal-
State-
Type.SUCCEEDED,
date:
date-
time_serializer,
re-
turn_type=int,
when_used=unless
none)],
url:
str,
**ex-
tra_data:
Any)
```

Any)

The withdrawal succeeded.

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

type: Literal[RevenueWithdrawalStateType.SUCCEEDED]

Type of the state, always 'succeeded'

```
date: DateTime
```

Date the withdrawal was completed in Unix time

url: str

An HTTPS URL that can be used to see transaction details

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

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

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

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 <code>aiogram.methods.answer_shipping_query.AnswerShippingQuery</code> will automatically fill method attributes:

• shipping_query_id

If you sent an invoice requesting a shipping address and the parameter *is_flexible* was specified, the Bot API will send an *aiogram.types.update.Update* with a *shipping_query* field to the bot. Use this method to reply to shipping queries. On success, True is returned.

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

Parameters

- **ok** Pass True if delivery to the specified address is possible and False if there are any problems (for example, if delivery to the specified address is not possible)
- **shipping_options** Required if *ok* is True. A JSON-serialized array of available shipping options.
- **error_message** Required if *ok* is False. Error message in human readable form that explains why it is impossible to complete the order (e.g. 'Sorry, delivery to your desired address is unavailable'). Telegram will display this message to the user.

Returns

 $\begin{tabular}{ll} instance & of & method & aiogram.methods.answer_shipping_query. \\ AnswerShippingQuery & \end{tabular}$

StarTransaction

class aiogram.types.star_transaction.**StarTransaction**(*, id: str, amount: int, date:

_datetime_serializer, return_type=int, when_used=unless - none)], nanostar_amount: $int \mid None = None, source:$ TransactionPartnerUser | TransactionPartnerChat | TransactionPartnerAffiliateProgram | TransactionPartnerFragment | TransactionPartnerTelegramAds | TransactionPartnerTelegramApi | TransactionPartnerOther | None = None, receiver: TransactionPartnerUser | TransactionPartnerChat | TransactionPartnerAffiliateProgram | TransactionPartnerFragment | TransactionPartnerTelegramAds | TransactionPartnerTelegramApi | TransactionPartnerOther | None = None,

**extra_data: Any)

Describes a Telegram Star transaction. Note that if the buyer initiates a chargeback with the payment provider from whom they acquired Stars (e.g., Apple, Google) following this transaction, the refunded Stars will be deducted from the bot's balance. This is outside of Telegram's control.

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

id: str

Unique identifier of the transaction. Coincides with the identifier of the original transaction for refund transactions. Coincides with *SuccessfulPayment.telegram_payment_charge_id* for successful incoming payments from users.

amount: int

Integer amount of Telegram Stars transferred by the transaction

date: DateTime

Date the transaction was created in Unix time

nanostar_amount: int | None

Optional. The number of 1/1000000000 shares of Telegram Stars transferred by the transaction; from 0 to 999999999

source: TransactionPartnerUnion | None

Optional. Source of an incoming transaction (e.g., a user purchasing goods or services, Fragment refunding a failed withdrawal). Only for incoming transactions

receiver: TransactionPartnerUnion | None

Optional. Receiver of an outgoing transaction (e.g., a user for a purchase refund, Fragment for a withdrawal). Only for outgoing transactions

StarTransactions

Contains a list of Telegram Star transactions.

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

transactions: list[StarTransaction]

The list of transactions

SuccessfulPayment

This object contains basic information about a successful payment. Note that if the buyer initiates a chargeback with the relevant payment provider following this transaction, the funds may be debited from your balance. This is outside of Telegram's control.

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

```
currency: str
```

Three-letter ISO 4217 currency code, or 'XTR' for payments in Telegram Stars

```
total_amount: int
```

Total price in the *smallest units* of the currency (integer, **not** float/double). For example, for a price of US\$ 1.45 pass amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies).

```
invoice_payload: str
```

Bot-specified invoice payload

telegram_payment_charge_id: str

Telegram payment identifier

provider_payment_charge_id: str

Provider payment identifier

```
subscription_expiration_date: int | None
```

Optional. Expiration date of the subscription, in Unix time; for recurring payments only

is_recurring: bool | None

Optional. True, if the payment is a recurring payment for a subscription

is_first_recurring: bool | None

Optional. True, if the payment is the first payment for a subscription

```
shipping_option_id: str | None
          Optional. Identifier of the shipping option chosen by the user
order_info: OrderInfo | None
          Optional. Order information provided by the user
```

TransactionPartner

class aiogram.types.transaction_partner.TransactionPartner(**extra_data: Any)

This object describes the source of a transaction, or its recipient for outgoing transactions. Currently, it can be one of

- aiogram.types.transaction_partner_user.TransactionPartnerUser
- aiogram.types.transaction_partner_chat.TransactionPartnerChat
- aiogram.types.transaction_partner_affiliate_program.TransactionPartnerAffiliateProgram
- aiogram.types.transaction_partner_fragment.TransactionPartnerFragment
- aiogram.types.transaction_partner_telegram_ads.TransactionPartnerTelegramAds
- aiogram.types.transaction_partner_telegram_api.TransactionPartnerTelegramApi
- aiogram.types.transaction_partner_other.TransactionPartnerOther

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

TransactionPartnerAffiliateProgram

class aiogram.types.transaction_partner_affiliate_program.TransactionPartnerAffiliateProgram(*,

```
Trans-
ac-
tion-
Part-
ner-
Type.AFFILL
com-
mis-
sion_per_mil
int,
spon-
sor_user:
User
None
None,
**ex-
tra data:
```

Any)

type: Lit-

eral[Transact

Describes the affiliate program that issued the affiliate commission received via this transaction.

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

type: Literal[TransactionPartnerType.AFFILIATE_PROGRAM]

Type of the transaction partner, always 'affiliate_program'

```
commission_per_mille: int
```

The number of Telegram Stars received by the bot for each 1000 Telegram Stars received by the affiliate program sponsor from referred users

```
sponsor_user: User | None
```

Optional. Information about the bot that sponsored the affiliate program

TransactionPartnerChat

Describes a transaction with a chat.

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

type: Literal[TransactionPartnerType.CHAT]

Type of the transaction partner, always 'chat'

chat: Chat

Information about the chat

gift: Gift | None

Optional. The gift sent to the chat by the bot

TransactionPartnerFragment

class aiogram.types.transaction_partner_fragment.TransactionPartnerFragment(*, type: Lit-

eral[TransactionPartnerType.FRAG

 $= Transaction \hbox{-}$

Partner-

Type.FRAGMENT,

with-

drawal_state:

RevenueWith-

drawalStatePend-

ing

RevenueWith-

drawalStateSuc-

ceeded

RevenueWith-

drawalState-

Failed | None =

None,

 $**extra_data:$

Any)

Describes a withdrawal transaction with Fragment.

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

type: Literal[TransactionPartnerType.FRAGMENT]

Type of the transaction partner, always 'fragment'

withdrawal_state: RevenueWithdrawalStateUnion | None

Optional. State of the transaction if the transaction is outgoing

TransactionPartnerOther

class aiogram.types.transaction_partner_other.TransactionPartnerOther(*, type: Lit-

eral[TransactionPartnerType.OTHER]

= Transaction Partner-

Type.OTHER, **extra_data: Any)

Describes a transaction with an unknown source or recipient.

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

type: Literal[TransactionPartnerType.OTHER]

Type of the transaction partner, always 'other'

TransactionPartnerTelegramAds

Describes a withdrawal transaction to the Telegram Ads platform.

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

type: Literal[TransactionPartnerType.TELEGRAM_ADS]

Type of the transaction partner, always 'telegram_ads'

TransactionPartnerTelegramApi

class aiogram.types.transaction_partner_telegram_api.TransactionPartnerTelegramApi(*, type:

eral[TransactionPartnerTy

=
TransactionPartnerType.TELEGRAM_API,
request_count:
int,
**extra data:

Any)

Lit-

Any)

Describes a transaction with payment for paid broadcasting.

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

type: Literal[TransactionPartnerType.TELEGRAM_API]

Type of the transaction partner, always 'telegram_api'

request_count: int

The number of successful requests that exceeded regular limits and were therefore billed

TransactionPartnerUser

```
class aiogram.types.transaction_partner_user.TransactionPartnerUser(*, type: Lit-
                                                                                 eral[TransactionPartnerType.USER]
                                                                                 = Transaction Partner-
                                                                                 Type. USER, user: User,
                                                                                 affiliate: AffiliateInfo |
                                                                                 None = None,
                                                                                 invoice_payload: str | None
                                                                                 = None.
                                                                                 subscription_period: int |
                                                                                 None = None, paid\_media:
                                                                                 list[PaidMediaPreview |
                                                                                 PaidMediaPhoto I
                                                                                 PaidMediaVideo | | None =
                                                                                 None,
                                                                                 paid_media_payload: str |
                                                                                 None = None, gift: Gift |
                                                                                 None = None,
                                                                                 **extra_data: Any)
     Describes a transaction with a user.
     Source: https://core.telegram.org/bots/api#transactionpartneruser
     type: Literal[TransactionPartnerType.USER]
          Type of the transaction partner, always 'user'
     user: User
          Information about the user
     affiliate: AffiliateInfo | None
          Optional. Information about the affiliate that received a commission via this transaction
     invoice_payload: str | None
          Optional. Bot-specified invoice payload
     subscription_period: int | None
          Optional. The duration of the paid subscription
     paid_media: list[PaidMediaUnion] | None
          Optional. Information about the paid media bought by the user
     paid_media_payload: str | None
          Optional. Bot-specified paid media payload
     gift: Gift | None
          Optional. The gift sent to the user by the bot
```

Stickers

Gift

This object represents a gift that can be sent by the bot.

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

id: str

Unique identifier of the gift

sticker: Sticker

The sticker that represents the gift

star_count: int

The number of Telegram Stars that must be paid to send the sticker

upgrade_star_count: int | None

Optional. The number of Telegram Stars that must be paid to upgrade the gift to a unique one

total_count: int | None

Optional. The total number of the gifts of this type that can be sent; for limited gifts only

remaining_count: int | None

Optional. The number of remaining gifts of this type that can be sent; for limited gifts only

Gifts

```
class aiogram.types.gifts.Gifts(*, gifts: list[Gift], **extra_data: Any)
```

This object represent a list of gifts.

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

gifts: list[Gift]
The list of gifts

InputSticker

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

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

sticker: InputFileUnion

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

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.

y_shift: float

Shift by Y-axis measured in heights of the mask scaled to the face size, from top to bottom. For example, 1.0 will place the mask just below the default mask position.

scale: float

Mask scaling coefficient. For example, 2.0 means double size.

Sticker

This object represents a sticker.

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

file_id: str

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

file_unique_id: str

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

type: str

Type of the sticker, currently one of 'regular', 'mask', 'custom_emoji'. The type of the sticker is independent from its format, which is determined by the fields *is_animated* and *is_video*.

width: int

Sticker width

height: int

Sticker height

is_animated: bool

True, if the sticker is animated

is video: bool

True, if the sticker is a video sticker

thumbnail: PhotoSize | None

Optional. Sticker thumbnail in the .WEBP or .JPG format

emoji: str | None

Optional. Emoji associated with the sticker

set_name: str | None

Optional. Name of the sticker set to which the sticker belongs

premium_animation: File | None

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

mask_position: MaskPosition | None

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

custom_emoji_id: str | None

Optional. For custom emoji stickers, unique identifier of the custom emoji

needs_repainting: bool | None

Optional. True, if the sticker must be repainted to a text color in messages, the color of the Telegram Premium badge in emoji status, white color on chat photos, or another appropriate color in other places

file_size: int | None

Optional. File size in bytes

$set_position_in_set(position: int, **kwargs: Any) \rightarrow SetStickerPositionInSet$

 $Shortcut \qquad for \qquad method \qquad aiogram.methods.set_sticker_position_in_set. \\ SetStickerPositionInSet \ will \ automatically \ fill \ method \ attributes:$

• 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

```
\begin{array}{ll} \text{instance} & \text{of} & \text{method} \\ & aiogram.\textit{methods.delete\_sticker\_from\_set.} \\ \\ \textit{DeleteStickerFromSet} \end{array}
```

StickerSet

This object represents a sticker set.

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

name: str

Sticker set name

title: str

Sticker set title

sticker_type: str

Type of stickers in the set, currently one of 'regular', 'mask', 'custom_emoji'

stickers: list[Sticker]

List of all set stickers

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> EncryptedPassportElement. See the Telegram Passport Documentation for a complete description of the data decryption and authentication processes.

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

data: str

Base64-encoded encrypted JSON-serialized data with unique user's payload, data hashes and secrets required for aiogram.types.encrypted_passport_element.EncryptedPassportElement decryption and authentication

hash: str

Base64-encoded data hash for data authentication

secret: str

Base64-encoded secret, encrypted with the bot's public RSA key, required for data decryption

EncryptedPassportElement

class aiogram.types.encrypted_passport_element.EncryptedPassportElement(*, type: str, hash: str,

data: str | None = *None, phone number:* $str \mid None = None$, email: str | None = None, files: list[PassportFile] | None = None, front side: PassportFile | *None* = None, reverse_side: PassportFile | None = None, selfie: PassportFile | *None* = *None*, translation: list/PassportFile/ None = None,**extra_data: Any)

Describes documents or other Telegram Passport elements shared with the bot by the user.

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

type: str

Element type. One of 'personal_details', 'passport', 'driver_license', 'identity_card', 'internal_passport', 'address', 'utility_bill', 'bank_statement', 'rental_agreement', 'passport_registration', 'temporary_registration', 'phone_number', 'email'.

hash: str

 $Base 64-encoded\ element\ hash\ for\ using\ in\ aiogram.\ types.passport_element_error_unspecified.$ PassportElementErrorUnspecified

data: str | None

Optional. Base64-encoded encrypted Telegram Passport element data provided by the user; available only for 'personal_details', 'passport', 'driver_license', 'identity_card', 'internal_passport' and 'address' types. Can be decrypted and verified using the accompanying aiogram.types.encrypted_credentials. EncryptedCredentials.

phone_number: str | None

Optional. User's verified phone number; available only for 'phone_number' type

email: str | None

Optional. User's verified email address; available only for 'email' type

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 aiogram.types.encrypted_credentials.EncryptedCredentials.

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 <code>aiogram.types.encrypted_credentials.EncryptedCredentials.</code>

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 bot

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
- $\bullet \ aiogram.types.passport_element_error_translation_files.PassportElementErrorTranslationFiles.PassportElementErrorTran$
- aiogram.types.passport_element_error_unspecified.PassportElementErrorUnspecified

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

PassportElementErrorDataField

class aiogram.types.passport_element_error_data_field.PassportElementErrorDataField(*,

```
Lit-
eral[PassportElementErr
Pass-
portEle-
mentEr-
rorType.DATA,
type:
str,
field name:
str.
data_hash:
str,
mes-
sage:
str,
**ex-
tra data:
Any)
```

source:

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

data_hash: str

Base64-encoded data hash

message: str
Error message

PassportElementErrorFile

```
\textbf{class} \ \texttt{aiogram.types.passport\_element\_error\_file.} \textbf{PassportElementErrorFile} (*, \textit{source: Lit-passportElementErrorFile}) (*, \textit{source: Lit-passpo
```

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'

file_hash: str

Base64-encoded file hash

message: str
Error message

PassportElementErrorFiles

```
class aiogram.types.passport_element_error_files.PassportElementErrorFiles(*, source: Lit-
```

```
eral[PassportElementErrorType.FILL
= PassportEle-
mentEr-
rorType.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'

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'

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'

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'

file_hash: str

Base64-encoded hash of the file with the selfie

message: str Error message

PassportElementErrorTranslationFile

 ${\bf class}\ {\bf aiogram.types.passport_element_error_translation_file. \textbf{PassportElementErrorTranslationFile}(*, aiogram.types.passport_element_error_translation_file.)$

PassportElementErrorType.T
type:
str,
file_hash.
str,
message:
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'

file_hash: str

Base64-encoded file hash

message: str Error message

PassportElementErrorTranslationFiles

class aiogram.types.passport_element_error_translation_files.PassportElementErrorTranslationFiles(*,

source
Literal[Po
=
PassportEle
mentEle
rorTyp
type:
str,
file_ha
list[str
message:
str,
**ex-

tra_da Any)

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'

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-
e-
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

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.

file_size: int
File size in bytes

file_date: DateTime

Unix time when the file was uploaded

Getting updates

Update

class aiogram.types.update.**Update**(*, update_id: int, message: Message | None = None, edited_message: Message | None = None, channel post: Message | None = None

Message | None = None, channel post: Message | None = None, edited channel post: Message | None = None, business connection: BusinessConnection | None = None, business_message: Message | None = None, edited business message: Message | None = None, deleted business messages: BusinessMessagesDeleted | None = None, message reaction: MessageReactionUpdated | None = None, message reaction count: MessageReactionCountUpdated | None = *None*, *inline_query*: InlineQuery | *None* = *None*, *chosen_inline_result*: ChosenInlineResult | *None* = *None*, *callback_query*: CallbackQuery | *None* = *None*, *shipping_query*: ShippingQuery | *None* = *None*, pre_checkout_query: PreCheckoutQuery | None = None, purchased_paid_media: PaidMediaPurchased | None = None, poll: Poll $| None = None, poll_answer: PollAnswer | None = None,$ *my_chat_member*: ChatMemberUpdated | *None* = *None*, *chat_member*: ChatMemberUpdated | *None* = *None*, *chat_join_request*: ChatJoinRequest | None = None, chat_boost: ChatBoostUpdated | None = None, removed chat boost: ChatBoostRemoved | None = None, **extra_data: Any)

This object represents an incoming update.

At most **one** of the optional parameters can be present in any given update.

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

update_id: int

The update's unique identifier. Update identifiers start from a certain positive number and increase sequentially. This identifier becomes especially handy if you're using webhooks, since it allows you to ignore repeated updates or to restore the correct update sequence, should they get out of order. If there are no new updates for at least a week, then identifier of the next update will be chosen randomly instead of sequentially.

message: Message | None

Optional. New incoming message of any kind - text, photo, sticker, etc.

edited_message: Message | None

Optional. New version of a message that is known to the bot and was edited. This update may at times be triggered by changes to message fields that are either unavailable or not actively used by your bot.

channel_post: Message | None

Optional. New incoming channel post of any kind - text, photo, sticker, etc.

edited_channel_post: Message | None

Optional. New version of a channel post that is known to the bot and was edited. This update may at times be triggered by changes to message fields that are either unavailable or not actively used by your bot.

business_connection: BusinessConnection | None

Optional. The bot was connected to or disconnected from a business account, or a user edited an existing connection with the bot

business_message: Message | None

Optional. New message from a connected business account

edited_business_message: Message | None

Optional. New version of a message from a connected business account

deleted_business_messages: BusinessMessagesDeleted | None

Optional. Messages were deleted from a connected business account

message_reaction: MessageReactionUpdated | None

Optional. A reaction to a message was changed by a user. The bot must be an administrator in the chat and must explicitly specify "message_reaction" in the list of *allowed_updates* to receive these updates. The update isn't received for reactions set by bots.

message_reaction_count: MessageReactionCountUpdated | None

Optional. Reactions to a message with anonymous reactions were changed. The bot must be an administrator in the chat and must explicitly specify "message_reaction_count" in the list of *allowed_updates* to receive these updates. The updates are grouped and can be sent with delay up to a few minutes.

inline_query: InlineQuery | None

Optional. New incoming inline query

chosen_inline_result: ChosenInlineResult | None

Optional. The result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback collecting for details on how to enable these updates for your bot.

callback_query: CallbackQuery | None

Optional. New incoming callback query

shipping_query: ShippingQuery | None

Optional. New incoming shipping query. Only for invoices with flexible price

pre_checkout_query: PreCheckoutQuery | None

Optional. New incoming pre-checkout query. Contains full information about checkout

purchased_paid_media: PaidMediaPurchased | None

Optional. A user purchased paid media with a non-empty payload sent by the bot in a non-channel chat

poll: Poll | None

Optional. New poll state. Bots receive only updates about manually stopped polls and polls, which are sent by the bot

poll_answer: PollAnswer | None

Optional. A user changed their answer in a non-anonymous poll. Bots receive new votes only in polls that were sent by the bot itself.

my_chat_member: ChatMemberUpdated | None

Optional. The bot's chat member status was updated in a chat. For private chats, this update is received only when the bot is blocked or unblocked by the user.

chat_member: ChatMemberUpdated | None

Optional. A chat member's status was updated in a chat. The bot must be an administrator in the chat and must explicitly specify "chat_member" in the list of *allowed_updates* to receive these updates.

chat_join_request: ChatJoinRequest | None

Optional. A request to join the chat has been sent. The bot must have the *can_invite_users* administrator right in the chat to receive these updates.

chat_boost: ChatBoostUpdated | None

Optional. A chat boost was added or changed. The bot must be an administrator in the chat to receive these updates.

removed_chat_boost: ChatBoostRemoved | None

Optional. A boost was removed from a chat. The bot must be an administrator in the chat to receive these updates.

property event_type: str

Detect update type If update type is unknown, raise UpdateTypeLookupError

Returns

```
property event: TelegramObject
```

exception aiogram.types.update.UpdateTypeLookupError

Update does not contain any known event type.

WebhookInfo

class aiogram.types.webhook_info.WebhookInfo(*, url: str, has_custom_certificate: bool,

pending_update_count: int, ip_address: str | None = None, last_error_date: _datetime_serializer, return_type=int, when_used=unless - none)] | None = None, last_error_message: str | None = None, last_synchronization_error_date: _datetime_serializer, return_type=int, when_used=unless - none)] | None = None, max_connections: int | None = None, allowed_updates: list[str] | None = None, **extra_data: Any)

Describes the current status of a webhook.

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

url: str

Webhook URL, may be empty if webhook is not set up

has_custom_certificate: bool

True, if a custom certificate was provided for webhook certificate checks

pending_update_count: int

Number of updates awaiting delivery

ip_address: str | None

Optional. Currently used webhook IP address

last_error_date: DateTime | None

Optional. Unix time for the most recent error that happened when trying to deliver an update via webhook

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

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.

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

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

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.

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

sticker_type: str | None

Type of stickers in the set, pass 'regular', 'mask', or 'custom_emoji'. By default, a regular sticker set is created.

needs_repainting: bool | None

Pass True if stickers in the sticker set must be repainted to the color of text when used in messages, the accent color if used as emoji status, white on chat photos, or another appropriate color based on context; for custom emoji sticker sets only

sticker_format: str | None

Format of stickers in the set, must be one of 'static', 'animated', 'video'

Deprecated since version API:7.2: https://core.telegram.org/bots/api-changelog#march-31-2024

Usage

As bot method

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

Method as object

Imports:

- 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

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.delete_sticker_from_set import DeleteStickerFromSet
- alias: from aiogram.methods import DeleteStickerFromSet

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

As bot method

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

Method as object

Imports:

- from aiogram.methods.delete_sticker_set import DeleteStickerSet
- alias: from aiogram.methods import DeleteStickerSet

With specific bot

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

As reply into Webhook in handler

```
return DeleteStickerSet(...)
```

getAvailableGifts

Returns: Gifts

class aiogram.methods.get_available_gifts.GetAvailableGifts(**extra_data: Any)

Returns the list of gifts that can be sent by the bot to users and channel chats. Requires no parameters. Returns a <code>aiogram.types.gifts.Gifts</code> object.

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

Usage

As bot method

```
result: Gifts = await bot.get_available_gifts(...)
```

Method as object

Imports:

- from aiogram.methods.get_available_gifts import GetAvailableGifts
- alias: from aiogram.methods import GetAvailableGifts

With specific bot

```
result: Gifts = await bot(GetAvailableGifts(...))
```

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.

Usage

As bot method

```
result: list[Sticker] = await bot.get_custom_emoji_stickers(...)
```

Method as object

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

Usage

As bot method

```
result: StickerSet = await bot.get_sticker_set(...)
```

Method as object

Imports:

- from aiogram.methods.get_sticker_set import GetStickerSet
- alias: from aiogram.methods import GetStickerSet

With specific bot

```
result: StickerSet = await bot(GetStickerSet(...))
```

removeChatVerification

Returns: bool

Removes verification from a chat that is currently verified on behalf of the organization represented by the bot. Returns True on success.

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

As bot method

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

Method as object

Imports:

- from aiogram.methods.remove_chat_verification import RemoveChatVerification
- alias: from aiogram.methods import RemoveChatVerification

With specific bot

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

As reply into Webhook in handler

```
return RemoveChatVerification(...)
```

removeUserVerification

Returns: bool

Removes verification from a user who is currently verified on behalf of the organization represented by the bot. Returns True on success.

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

user_id: int

Unique identifier of the target user

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.remove_user_verification import RemoveUserVerification
- alias: from aiogram.methods import RemoveUserVerification

With specific bot

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

As reply into Webhook in handler

```
return RemoveUserVerification(...)
```

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 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.

As bot method

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

Method as object

Imports:

- from aiogram.methods.replace_sticker_in_set import ReplaceStickerInSet
- alias: from aiogram.methods import ReplaceStickerInSet

With specific bot

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

As reply into Webhook in handler

```
        return
        ReplaceStickerInSet(...)
```

sendGift

Returns: bool

Sends a gift to the given user or channel chat. The gift can't be converted to Telegram Stars by the receiver. Returns True on success.

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

```
gift_id: str
```

Identifier of the gift

```
user_id: int | None
```

Required if chat_id is not specified. Unique identifier of the target user who will receive the gift.

chat_id: ChatIdUnion | None

Required if *user_id* is not specified. Unique identifier for the chat or username of the channel (in the format @channelusername) that will receive the gift.

```
pay_for_upgrade: bool | None
```

Pass True to pay for the gift upgrade from the bot's balance, thereby making the upgrade free for the receiver

text: str | None

Text that will be shown along with the gift; 0-255 characters

text_parse_mode: str | None

Mode for parsing entities in the text. See formatting options for more details. Entities other than 'bold', 'italic', 'underline', 'strikethrough', 'spoiler', and 'custom_emoji' are ignored.

```
text_entities: list[MessageEntity] | None
```

A JSON-serialized list of special entities that appear in the gift text. It can be specified instead of *text_parse_mode*. Entities other than 'bold', 'italic', 'underline', 'strikethrough', 'spoiler', and 'custom_emoji' are ignored.

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.send_gift import SendGift
- alias: from aiogram.methods import SendGift

With specific bot

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

As reply into Webhook in handler

```
return SendGift(...)
```

sendSticker

Returns: Message

```
class aiogram.methods.send_sticker.SendSticker(*, chat id: int | str, sticker: str |
                                                          ~aiogram.types.input_file.InputFile,
                                                          business connection id: str \mid None = None,
                                                          message_thread_id: int | None = None, emoji: str |
                                                          None = None, disable notification: bool | None =
                                                          None, protect content: bool |
                                                          ~aiogram.client.default.Default | None =
                                                          <Default('protect content')>, allow paid broadcast:
                                                          bool | None = None, message_effect_id: str | None =
                                                          None, reply_parameters:
                                                          ~aiogram.types.reply_parameters.ReplyParameters |
                                                          None = None, reply_markup:
                                                          ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                          | ~aiogram.types.force reply.ForceReply | None =
                                                          None, allow_sending_without_reply: bool | None =
                                                          None, reply to message id: int | None = None,
                                                          **extra_data: ~typing.Any)
```

Use this method to send static .WEBP, animated .TGS, or video .WEBM stickers. On success, the sent aiogram. types.message is returned.

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

sticker: InputFileUnion

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.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

reply_markup: ReplyMarkupUnion | 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()

- aiogram.types.inaccessible_message.InaccessibleMessage.answer_sticker()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_sticker()

setCustomEmojiStickerSetThumbnail

Returns: bool

class aiogram.methods.set_custom_emoji_sticker_set_thumbnail.SetCustomEmojiStickerSetThumbnail(*,

name:
 str,
 cus tom_emoji
 str
 |
 None
 =
 None,
 **ex-

tra_data:
Any)

Use this method to set the thumbnail of a custom emoji sticker set. Returns True on success.

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

name: str

Sticker set name

custom_emoji_id: str | None

Custom emoji identifier of a sticker from the sticker set; pass an empty string to drop the thumbnail and use the first sticker as the thumbnail.

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.set_custom_emoji_sticker_set_thumbnail import SetCustomEmojiStickerSetThumbnail
- alias: from aiogram.methods import SetCustomEmojiStickerSetThumbnail

With specific bot

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

As reply into Webhook in handler

```
        return
        SetCustomEmojiStickerSetThumbnail(...)
```

setStickerEmojiList

Returns: bool

Use this method to change the list of emoji assigned to a regular or custom emoji sticker. The sticker must belong to a sticker set created by the bot. Returns True on success.

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

sticker: str

File identifier of the sticker

emoji_list: list[str]

A JSON-serialized list of 1-20 emoji associated with the sticker

Usage

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

keywords: list[str] | None

A JSON-serialized list of 0-20 search keywords for the sticker with total length of up to 64 characters

Usage

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

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.

Usage

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

position: int

New sticker position in the set, zero-based

Usage

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

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: InputFileUnion | None

A .WEBP or .PNG image with the thumbnail, must be up to 128 kilobytes in size and have a width and height of exactly 100px, or a .TGS animation with a thumbnail up to 32 kilobytes in size (see https://core.telegram.org/stickers#animation-requirements https://core.telegram.org/stickers#animation-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-requirements https://core.telegram.o

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

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 https://core.telegram.org/stickers information on Sending Files »

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

With specific bot

```
result: File = await bot(UploadStickerFile(...))
```

verifyChat

Returns: bool

Verifies a chat on behalf of the organization which is represented by the bot. Returns True on success.

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

custom_description: str | None

Custom description for the verification; 0-70 characters. Must be empty if the organization isn't allowed to provide a custom verification description.

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.verify_chat import VerifyChat
- alias: from aiogram.methods import VerifyChat

With specific bot

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

As reply into Webhook in handler

```
return VerifyChat(...)
```

verifyUser

Returns: bool

Verifies a user on behalf of the organization which is represented by the bot. Returns True on success.

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

user_id: int

Unique identifier of the target user

custom_description: str | None

Custom description for the verification; 0-70 characters. Must be empty if the organization isn't allowed to provide a custom verification description.

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.verify_user import VerifyUser
- alias: from aiogram.methods import VerifyUser

With specific bot

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

As reply into Webhook in handler

```
return VerifyUser(...)
```

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*.

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.

Usage

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: ChatIdUnion

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

Usage

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: ChatIdUnion

Unique identifier for the target group or username of the target supergroup or channel (in the format @channelusername)

user_id: int

Unique identifier of the target user

until date: DateTimeUnion | 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.

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

sender_chat_id: int

Unique identifier of the target sender chat

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

As bot method

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

Method as object

Imports:

- from aiogram.methods.close import Close
- alias: from aiogram.methods import Close

With specific bot

```
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: ChatIdUnion

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

As bot method

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

Method as object

Imports:

- from aiogram.methods.close_forum_topic import CloseForumTopic
- alias: from aiogram.methods import CloseForumTopic

With specific bot

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

As reply into Webhook in handler

```
return CloseForumTopic(...)
```

closeGeneralForumTopic

Returns: bool

Use this method to close an open 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can_manage_topics* administrator rights. Returns True on success.

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

chat_id: ChatIdUnion

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

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.close_general_forum_topic import CloseGeneralForumTopic
- alias: from aiogram.methods import CloseGeneralForumTopic

With specific bot

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

As reply into Webhook in handler

```
return CloseGeneralForumTopic(...)
```

copyMessage

```
Returns: MessageId
```

```
class aiogram.methods.copy_message.CopyMessage(*, chat_id: int | str, from_chat_id: int | str,
                                                          message_id: int, message_thread_id: int | None =
                                                          None, video start timestamp: ~datetime.datetime |
                                                          \simdatetime.timedelta | int | None = None, caption: str |
                                                          None = None, parse\_mode: str
                                                          ~aiogram.client.default.Default | None =
                                                          <Default('parse_mode')>, caption_entities:
                                                          list[~aiogram.types.message entity.MessageEntity] |
                                                          None = None, show_caption_above_media: bool |
                                                          ~aiogram.client.default.Default | None =
                                                          <Default('show_caption_above_media')>,
                                                          disable\_notification: bool \mid None = None,
                                                          protect_content: bool | ~aiogram.client.default.Default
                                                          |None = < Default('protect\_content')>,
                                                          allow_paid_broadcast: bool | None = None,
                                                          reply_parameters:
                                                          ~aiogram.types.reply_parameters.ReplyParameters |
                                                          None = None, reply markup:
                                                          ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                          | ~aiogram.types.force_reply.ForceReply | None =
                                                          None, allow sending without reply: bool | None =
                                                          None, reply\_to\_message\_id: int | None = None,
```

Use this method to copy messages of any kind. Service messages, paid media messages, giveaway messages, giveaway winners messages, and invoice messages can't be copied. A quiz aiogram.methods.poll.Poll can be copied only if the value of the field correct_option_id is known to the bot. The method is analogous to

**extra_data: ~typing.Any)

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 <code>success</code>

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

from_chat_id: ChatIdUnion

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

video_start_timestamp: DateTimeUnion | None

New start timestamp for the copied video in the message

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

show_caption_above_media: bool | Default | None

Pass True, if the caption must be shown above the message media. Ignored if a new caption isn't specified.

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

As bot method

```
result: MessageId = await bot.copy_message(...)
```

Method as object

Imports:

- from aiogram.methods.copy_message import CopyMessage
- alias: from aiogram.methods import CopyMessage

With specific bot

```
result: MessageId = await bot(CopyMessage(...))
```

As reply into Webhook in handler

```
return CopyMessage(...)
```

As shortcut from received object

aiogram.types.message.Message.copy_to()

copyMessages

Returns: list[MessageId]

```
class aiogram.methods.copy_messages.CopyMessages(*, chat_id: int | str, from_chat_id: int | str, message_ids: list[int], message_thread_id: int | None = None, disable_notification: bool | None = None, protect_content: bool | None = None, remove_caption: bool | None = None, **extra_data: Any)
```

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

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

chat_id: int | str

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

from_chat_id: int | str

Unique identifier for the chat where the original messages were sent (or channel username in the format @channelusername)

message_ids: list[int]

A JSON-serialized list of 1-100 identifiers of messages in the chat *from_chat_id* to copy. The identifiers must be specified in a strictly increasing order.

message_thread_id: int | None

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

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: ChatIdUnion

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: DateTimeUnion | 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.create_chat_invite_link(...)
```

Method as object

Imports:

- from aiogram.methods.create_chat_invite_link import CreateChatInviteLink
- alias: from aiogram.methods import CreateChatInviteLink

With specific bot

```
result: ChatInviteLink = await bot(CreateChatInviteLink(...))
```

As reply into Webhook in handler

```
return CreateChatInviteLink(...)
```

As shortcut from received object

aiogram.types.chat.Chat.create_invite_link()

createChatSubscriptionInviteLink

Returns: ChatInviteLink

 $\textbf{class} \ \texttt{aiogram.methods.create_chat_subscription_invite_link.} \textbf{CreateChatSubscriptionInviteLink} (*, \texttt{aiogram.methods.create_chat_subscription_invite_link}) (*, \texttt{aiogram.methods.create_chat_subscription_invite_chat_subscription_invite_chat_subscription_invite_chat_subscription_invite_chat_subscription_invite_chat_subscription_invite_chat_subscription_invite_chat_subscription_invite_cha$

```
chat_id:
int
str,
sub-
scrip-
tion_period:
date-
time
timedelta
int,
sub-
scrip-
tion_price:
int,
name:
str
None
None,
**ex-
tra_data:
Any)
```

Use this method to create a subscription invite link for a channel chat. The bot must have the <code>can_invite_users</code> administrator rights. The link can be edited using the method <code>aiogram.methods.edit_chat_subscription_invite_link.EditChatSubscriptionInviteLink</code> or revoked using the method <code>aiogram.methods.revoke_chat_invite_link.RevokeChatInviteLink</code>. Returns the new invite link as a <code>aiogram.types.chat_invite_link.ChatInviteLink</code> object.

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

chat_id: ChatIdUnion

Unique identifier for the target channel chat or username of the target channel (in the format @channelusername)

subscription_period: DateTimeUnion

The number of seconds the subscription will be active for before the next payment. Currently, it must always be 2592000 (30 days).

subscription_price: int

The amount of Telegram Stars a user must pay initially and after each subsequent subscription period to be a member of the chat; 1-2500

name: str | None

Invite link name; 0-32 characters

Usage

As bot method

```
result: ChatInviteLink = await bot.create_chat_subscription_invite_link(...)
```

Method as object

Imports:

- from aiogram.methods.create_chat_subscription_invite_link import CreateChatSubscriptionInviteLink
- alias: from aiogram.methods import CreateChatSubscriptionInviteLink

With specific bot

```
result: ChatInviteLink = await bot(CreateChatSubscriptionInviteLink(...))
```

As reply into Webhook in handler

```
return CreateChatSubscriptionInviteLink(...)
```

createForumTopic

Returns: ForumTopic

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

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

chat_id: ChatIdUnion

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

name: str

Topic name, 1-128 characters

icon_color: int | None

Color of the topic icon in RGB format. Currently, must be one of 7322096 (0x6FB9F0), 16766590 (0xFFD67E), 13338331 (0xCB86DB), 9367192 (0x8EEE98), 16749490 (0xFF93B2), or 16478047 (0xFB6F5F)

icon_custom_emoji_id: str | None

Unique identifier of the custom emoji shown as the topic icon. Use <code>aiogram.methods.get_forum_topic_icon_stickers.GetForumTopicIconStickers</code> to get all allowed custom emoji identifiers.

Usage

As bot method

```
result: ForumTopic = await bot.create_forum_topic(...)
```

Method as object

Imports:

- from aiogram.methods.create_forum_topic import CreateForumTopic
- alias: from aiogram.methods import CreateForumTopic

With specific bot

```
result: ForumTopic = await bot(CreateForumTopic(...))
```

```
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: ChatIdUnion

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

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(...))
```

```
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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Usage

As bot method

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

Method as object

Imports:

396

- from aiogram.methods.delete_chat_photo import DeleteChatPhoto
- alias: from aiogram.methods import DeleteChatPhoto

With specific bot

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

```
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: ChatIdUnion

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

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.delete_chat_sticker_set import DeleteChatStickerSet
- alias: from aiogram.methods import DeleteChatStickerSet

With specific bot

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

```
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: ChatIdUnion

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

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

 $\textbf{class} \ a iogram. \texttt{methods.delete_my_commands.DeleteMyCommands}(*, \textit{scope}: BotCommandScopeDefault | \texttt{methods.delete_my_commands}(*, \textit{scope}: BotCommandScopeDefault)(*, \texttt{methods.delete_my_commands}(*, \texttt{methods.delete_my_$

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: BotCommandScopeUnion | None

A JSON-serialized object, describing scope of users for which the commands are relevant. Defaults to aiogram.types.bot_command_scope_default.BotCommandScopeDefault.

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: ChatIdUnion

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

expire_date: DateTimeUnion | None

Point in time (Unix timestamp) when the link will expire

```
member_limit: int | None
```

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

```
creates_join_request: bool | None
```

True, if users joining the chat via the link need to be approved by chat administrators. If True, *member_limit* can't be specified

Usage

As bot method

```
result: ChatInviteLink = await bot.edit_chat_invite_link(...)
```

Method as object

Imports:

- from aiogram.methods.edit_chat_invite_link import EditChatInviteLink
- alias: from aiogram.methods import EditChatInviteLink

With specific bot

```
result: ChatInviteLink = await bot(EditChatInviteLink(...))
```

As reply into Webhook in handler

```
return EditChatInviteLink(...)
```

As shortcut from received object

• aiogram.types.chat.Chat.edit_invite_link()

editChatSubscriptionInviteLink

Returns: ChatInviteLink

 ${\bf class}\ {\bf aiogram.methods.edit_chat_subscription_invite_link.} {\bf EditChatSubscriptionInviteLink}(*,$

```
chat_id:
int
|
str,
in-
vite_link:
str,
name:
str
|
None
=
None,
**ex-
tra_data:
Any)
```

Use this method to edit a subscription invite link created by the bot. The bot must have the <code>can_invite_users</code> 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#editchatsubscriptioninvitelink

chat_id: ChatIdUnion

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

Usage

As bot method

```
result: ChatInviteLink = await bot.edit_chat_subscription_invite_link(...)
```

Method as object

Imports:

- from aiogram.methods.edit_chat_subscription_invite_link import EditChatSubscriptionInviteLink
- alias: from aiogram.methods import EditChatSubscriptionInviteLink

With specific bot

```
result: ChatInviteLink = await bot(EditChatSubscriptionInviteLink(...))
```

As reply into Webhook in handler

```
        return EditChatSubscriptionInviteLink(...)
```

editForumTopic

```
Returns: bool
```

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

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

chat_id: ChatIdUnion

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

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

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.edit_forum_topic import EditForumTopic
- alias: from aiogram.methods import EditForumTopic

With specific bot

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

As reply into Webhook in handler

```
return EditForumTopic(...)
```

editGeneralForumTopic

Returns: bool

Use this method to edit the name of the 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can_manage_topics* administrator rights. Returns True on success.

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

chat_id: ChatIdUnion

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

name: str

New topic name, 1-128 characters

Usage

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Usage

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

from_chat_id: ChatIdUnion

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

```
video_start_timestamp: DateTimeUnion | None
```

New start timestamp for the forwarded video in the message

```
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.

```
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

Usage

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

Usage

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: ChatIdUnion

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

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[ResultChatMemberUnion]

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: ChatIdUnion

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

Usage

As bot method

```
result: list[ResultChatMemberUnion] = await bot.get_chat_administrators(...)
```

Method as object

Imports:

- from aiogram.methods.get_chat_administrators import GetChatAdministrators
- \bullet alias: from alogram.methods import GetChatAdministrators

With specific bot

```
result: list[ResultChatMemberUnion] = await bot(GetChatAdministrators(...))
```

As shortcut from received object

• aiogram.types.chat.Chat.get_administrators()

getChatMember

Returns: ResultChatMemberUnion

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 <code>aiogram.types.chat_member.ChatMember</code> object on success.

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

chat_id: ChatIdUnion

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

user_id: int

Unique identifier of the target user

Usage

As bot method

```
result: ResultChatMemberUnion = 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

```
result: ResultChatMemberUnion = await bot(GetChatMember(...))
```

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: ChatIdUnion

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

Usage

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: ResultMenuButtonUnion

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

Usage

As bot method

```
result: ResultMenuButtonUnion = 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: ResultMenuButtonUnion = 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

Usage

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

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

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.get_me import GetMe
- alias: from aiogram.methods import GetMe

With specific bot

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

getMyCommands

Returns: list[BotCommand]

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

BotCommandScopeAllPrivateChats | BotCommandScopeAllGroupChats |

BotCommandScopeAllChatAdministrators |

BotCommandScopeChat |

BotCommandScopeChatAdministrators |

BotCommandScopeChatMember | *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 returned.

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

scope: BotCommandScopeUnion | None

A JSON-serialized object, describing scope of users. Defaults to aiogram.types.bot_command_scope_default.BotCommandScopeDefault.

```
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

 ${\bf class}\ aiogram. {\tt methods.get_my_default_administrator_rights}. {\tt GetMyDefaultAdministratorRights}(^*,$

```
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.

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

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

Usage

As bot method

```
result: BotName = await bot.get_my_name(...)
```

Method as object

Imports:

- from aiogram.methods.get_my_name import GetMyName
- alias: from aiogram.methods import GetMyName

With specific bot

```
result: BotName = await bot(GetMyName(...))
```

getMyShortDescription

Returns: BotShortDescription

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

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

language_code: str | None

A two-letter ISO 639-1 language code or an empty string

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)

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

```
class aiogram.methods.get_user_profile_photos.GetUserProfilePhotos(*, user_id: int, offset: int | None = None, limit: int | None = None, **extra_data: Any)
```

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.

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: ChatIdUnion

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

Usage

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: ChatIdUnion

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

Usage

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(...))
```

```
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

Usage

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(...))
```

```
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: ChatIdUnion

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

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be pinned

disable_notification: bool | None

Pass True if it is not necessary to send a notification to all chat members about the new pinned message. Notifications are always disabled in channels and private chats.

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.pin_chat_message import PinChatMessage
- alias: from aiogram.methods import PinChatMessage

With specific bot

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

As reply into Webhook in handler

```
return PinChatMessage(...)
```

As shortcut from received object

- aiogram.types.chat.Chat.pin_message()
- aiogram.types.message.Message.pin()

promoteChatMember

Returns: bool

class aiogram.methods.promote_chat_member.PromoteChatMember(*, chat_id: int | str, user_id: int,

is anonymous: $bool \mid None = None$, can_manage_chat: bool | None = None, can delete messages: bool

None = None,

can_manage_video_chats: bool | $None = None, can_restrict_members:$

 $bool \mid None = None,$

can_promote_members: bool | None = None, can_change_info: bool | None = None, can_invite_users: bool | None = None, can_post_stories: bool | None

= None, can_edit_stories: bool | None = None, can_delete_stories: bool |

 $None = None, can_post_messages:$

 $bool \mid None = None,$

can_edit_messages: bool | None = None, can_pin_messages: bool | None = None, can_manage_topics: bool |

 $None = None, **extra_data: Any)$

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

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

chat_id: ChatIdUnion

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

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: ChatIdUnion

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

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: ChatIdUnion

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

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: ChatIdUnion

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

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.

until_date: DateTimeUnion | 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: ChatIdUnion

Unique identifier of the target chat or username of the target channel (in the format @channelusername)

invite_link: str

The invite link to revoke

Usage

As bot method

```
result: ChatInviteLink = await bot.revoke_chat_invite_link(...)
```

Method as object

Imports:

- from aiogram.methods.revoke_chat_invite_link import RevokeChatInviteLink
- alias: from aiogram.methods import RevokeChatInviteLink

With specific bot

```
result: ChatInviteLink = await bot(RevokeChatInviteLink(...))
```

As reply into Webhook in handler

```
return RevokeChatInviteLink(...)
```

As shortcut from received object

• aiogram.types.chat.Chat.revoke_invite_link()

sendAnimation

```
Returns: Message
```

```
class aiogram.methods.send_animation.SendAnimation(*, chat_id: int | str, animation: str |
                                                                 ~aiogram.types.input_file.InputFile,
                                                                 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:
                                                                 list[~aiogram.types.message entity.MessageEntity]
                                                                 | None = None, show_caption_above_media: bool
                                                                 | ~aiogram.client.default.Default | None =
                                                                 <Default('show_caption_above_media')>,
                                                                 has\_spoiler: bool \mid None = None,
                                                                 disable\_notification: bool \mid None = None,
                                                                 protect_content: bool |
                                                                 ~aiogram.client.default.Default | None =
                                                                 < Default('protect_content')>,
                                                                 allow\_paid\_broadcast: bool \mid None = None,
                                                                 message\_effect\_id: str \mid None = None,
                                                                 reply_parameters:
                                                                 ~aiogram.types.reply parameters.ReplyParameters
                                                                 | None = None, reply_markup:
                                                                 ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                                 ~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup
                                                                 ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                                 | ~aiogram.types.force_reply.ForceReply | None =
                                                                 None, allow_sending_without_reply: bool | None
                                                                 = None, reply_to_message_id: int | None = None,
                                                                 **extra_data: ~typing.Any)
```

Use this method to send animation files (GIF or H.264/MPEG-4 AVC video without sound). On success, the sent <code>aiogram.types.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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

animation: InputFileUnion

Animation to send. Pass a file_id as String to send an animation that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get an animation from the Internet, or upload a new animation using multipart/form-data. *More information on Sending Files* »

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

message_thread_id: int | None

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

duration: int | None

Duration of sent animation in seconds

width: int | None Animation width

height: int | None

Animation height

thumbnail: InputFile | None

Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320. Ignored if the file is not uploaded using multipart/form-data. Thumbnails can't be reused and can be only uploaded as a new file, so you can pass 'attach://<file_attach_name>' if the thumbnail was uploaded using multipart/form-data under <file_attach_name>. *More information on Sending Files* »

caption: str | None

Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters after entities parsing

parse_mode: str | Default | None

Mode for parsing entities in the animation caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Pass True, if the caption must be shown above the message media

has_spoiler: bool | None

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

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_animation(...)
```

Method as object

Imports:

- from aiogram.methods.send_animation import SendAnimation
- alias: from aiogram.methods import SendAnimation

With specific bot

```
result: Message = await bot(SendAnimation(...))
```

As reply into Webhook in handler

```
return SendAnimation(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_animation()
- aiogram.types.message.Message.reply_animation()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_animation()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_animation_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_animation()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_animation()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_animation()

sendAudio

```
Returns: Message
```

```
class aiogram.methods.send_audio.SendAudio(*, chat_id: int | str, audio: str |
                                                        ~aiogram.types.input_file.InputFile,
                                                       business_connection_id: str | None = None,
                                                       message_thread_id: int | None = None, caption: str | None =
                                                       None, parse_mode: str | ~aiogram.client.default.Default |
                                                       None = \langle Default('parse\ mode') \rangle, caption entities:
                                                       list[~aiogram.types.message_entity.MessageEntity] | None =
                                                       None, duration: int \mid None = None, performer: str \mid 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')>, allow_paid_broadcast: bool |
                                                       None = None, message\_effect\_id: str | None = None,
                                                       reply_parameters:
                                                        ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                        None, reply_markup:
                                                        ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                        ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                       \hbox{$\sim$aiogram.types.reply\_keyboard\_remove.} Reply Keyboard Remove
                                                       | \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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

audio: InputFileUnion

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

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | 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()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_audio()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_audio()

sendChatAction

Returns: bool

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

Example: The ImageBot needs some time to process a request and upload the image. Instead of sending a text message along the lines of 'Retrieving image, please wait...', the bot may use <code>aiogram.methods.send_chat_action.SendChatAction</code> with <code>action = upload_photo</code>. The user will see a 'sending photo' status for the bot.

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

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

chat_id: ChatIdUnion

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.

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:

438

- from aiogram.methods.send_chat_action import SendChatAction
- alias: from aiogram.methods import SendChatAction

With specific bot

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

As reply into Webhook in handler

```
return SendChatAction(...)
```

As shortcut from received object

• aiogram.types.chat.Chat.do()

sendContact

```
Returns: Message
```

```
class aiogram.methods.send_contact.SendContact(*, chat id: int | str, phone number: str, first name:
                                                          str, business_connection_id: str | None = None,
                                                          message_thread_id: int | None = None, last_name: str |
                                                          None = None, vcard: str \mid None = None,
                                                           disable\_notification: bool \mid None = None,
                                                          protect_content: bool | ~aiogram.client.default.Default
                                                          | None = < Default('protect_content')>,
                                                           allow\_paid\_broadcast: bool \mid None = None,
                                                          message\_effect\_id: str \mid None = None,
                                                           reply_parameters:
                                                           ~aiogram.types.reply_parameters.ReplyParameters |
                                                          None = None, reply\_markup:
                                                           ~aiogram.types.inline keyboard markup.InlineKeyboardMarkup
                                                           ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                           ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                           | ~aiogram.types.force reply.ForceReply | None =
                                                           None, allow_sending_without_reply: bool | None =
                                                          None, reply\_to\_message\_id: int | None = None,
                                                           **extra_data: ~typing.Any)
```

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

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

chat_id: ChatIdUnion

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: strContact'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

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_contact(...)
```

Method as object

Imports:

- from aiogram.methods.send_contact import SendContact
- alias: from aiogram.methods import SendContact

With specific bot

```
result: Message = await bot(SendContact(...))
```

As reply into Webhook in handler

```
return SendContact(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_contact()
- aiogram.types.message.Message.reply_contact()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_contact()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_contact_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_contact()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_contact()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_contact()

sendDice

Returns: Message

```
class aiogram.methods.send_dice.SendDice(*, chat_id: int | str, business_connection_id: str | None =
                                                  None, message_thread_id: int | None = None, emoji: str | None
                                                  = None, disable notification: bool | None = None,
                                                  protect content: bool | ~aiogram.client.default.Default | None
                                                  = < Default('protect_content')>, allow_paid_broadcast: bool |
                                                  None = None, message\_effect\_id: str | None = None,
                                                  reply_parameters:
                                                  ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                  None, reply_markup:
                                                  ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                  ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                  ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                  | \sim aiogram.types.force\_reply.ForceReply | None = None,
                                                  allow\_sending\_without\_reply: bool | None = None,
                                                  reply_to_message_id: int | None = None, **extra_data:
                                                  ~typing.Any)
```

Use this method to send 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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

message_thread_id: int | None

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

emoji: str | None

Emoji on which the dice throw animation is based. Currently, must be one of ", ", ", ", or ". Dice can have values 1-6 for ", " and ", values 1-5 for " and ", and values 1-64 for ". Defaults to "

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | 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()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_dice()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_dice()

sendDocument

Returns: Message

```
class aiogram.methods.send_document.SendDocument(*, chat id: int | str, document: str |
                                                              ~aiogram.types.input_file.InputFile,
                                                              business connection id: str \mid None = None,
                                                             message_thread_id: int | None = None, thumbnail:
                                                              \simaiogram.types.input file.InputFile | None = None,
                                                              caption: str \mid None = None, parse mode: str \mid
                                                              ~aiogram.client.default.Default | None =
                                                              <Default('parse mode')>, caption entities:
                                                              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')>,
                                                              allow_paid_broadcast: bool | None = None,
                                                              message\_effect\_id: str \mid None = None,
                                                              reply_parameters:
                                                              ~aiogram.types.reply parameters.ReplyParameters |
                                                             None = None, reply markup:
                                                              ~aiogram.types.inline keyboard markup.InlineKeyboardMarkup
                                                              ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                              ~aiogram.types.reply keyboard remove.ReplyKeyboardRemove
                                                             | ~aiogram.types.force_reply.ForceReply | None =
                                                             None, allow sending without reply: bool | None =
                                                              None, reply_to_message_id: int \mid None = None,
```

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.

**extra_data: ~typing.Any)

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

document: InputFileUnion

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

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_document(...)
```

Method as object

Imports:

- from aiogram.methods.send_document import SendDocument
- alias: from aiogram.methods import SendDocument

With specific bot

```
result: Message = await bot(SendDocument(...))
```

As reply into Webhook in handler

```
return SendDocument(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_document()
- aiogram.types.message.Message.reply_document()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_document()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_document_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_document()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_document()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_document()

sendLocation

Returns: Message

```
class aiogram.methods.send_location.SendLocation(*, chat id: int | str, latitude: float, longitude: float,
                                                              business\_connection\_id: str \mid None = None,
                                                              message thread id: int | None = None,
                                                              horizontal\_accuracy: float \mid None = None,
                                                              live period: int | None = None, heading: int | None
                                                              = None, proximity alert radius: int | None = None,
                                                              disable notification: bool \mid None = None,
                                                              protect content: bool |
                                                              ~aiogram.client.default.Default | None =
                                                              <Default('protect_content')>,
                                                              allow\_paid\_broadcast: bool \mid None = None,
                                                              message\_effect\_id: str \mid None = None,
                                                              reply_parameters:
                                                              ~aiogram.types.reply_parameters.ReplyParameters |
                                                              None = None, reply_markup:
                                                              ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                              ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                              ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                              | ~aiogram.types.force_reply.ForceReply | None =
                                                              None, allow_sending_without_reply: bool | None =
                                                              None, reply_to_message_id: int \mid 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: ChatIdUnion

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 0x7FFFFFFF 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.

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
reply_markup: ReplyMarkupUnion | None
```

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

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

```
reply_to_message_id: int | None
```

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

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

Usage

As bot method

```
result: Message = await bot.send_location(...)
```

Method as object

Imports:

- from aiogram.methods.send_location import SendLocation
- alias: from aiogram.methods import SendLocation

With specific bot

```
result: Message = await bot(SendLocation(...))
```

As reply into Webhook in handler

```
return SendLocation(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_location()
- aiogram.types.message.Message.reply_location()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_location()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_location_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_location()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_location()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_location()

sendMediaGroup

Returns: list[Message]

class aiogram.methods.send_media_group.SendMediaGroup(*, chat_id: int | str, media:

```
list[~aiogram.types.input_media_audio.InputMediaAudio
~aiogram.types.input media document.InputMediaDocument
~aiogram.types.input_media_photo.InputMediaPhoto
~aiogram.types.input_media_video.InputMediaVideo],
business_connection_id: str | None = None,
message\_thread\_id: int \mid None = None,
disable\_notification: bool \mid None = None,
protect_content: bool |
\simaiogram.client.default.Default | None =
< Default('protect_content')>,
allow_paid_broadcast: bool | None = None,
message\_effect\_id: str \mid None = None,
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: \sim typing.Any)
```

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

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

chat id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

media: list[MediaUnion]

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.

```
protect_content: bool | Default | None
```

Protects the contents of the sent messages from forwarding and saving

```
allow_paid_broadcast: bool | None
```

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

```
message_effect_id: str | None
```

Unique identifier of the message effect to be added to the message; for private chats only

```
reply_parameters: ReplyParameters | None
```

Description of the message to reply to

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

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

```
reply_to_message_id: int | None
```

If the messages are a reply, ID of the original message

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

Usage

As bot method

```
result: list[Message] = await bot.send_media_group(...)
```

Method as object

Imports:

- from aiogram.methods.send_media_group import SendMediaGroup
- alias: from aiogram.methods import SendMediaGroup

With specific bot

```
result: list[Message] = await bot(SendMediaGroup(...))
```

As reply into Webhook in handler

```
return SendMediaGroup(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_media_group()
- aiogram.types.message.Message.reply_media_group()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_media_group()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_media_group_pm()
- $\bullet \ aiogram.types.chat_member_updated.ChatMemberUpdated.answer_media_group()\\$
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_media_group()
- $\bullet \ aiogram.types.inaccessible_message.InaccessibleMessage.reply_media_group()\\$

sendMessage

Returns: Message

```
class aiogram.methods.send_message.SendMessage(*, chat id: int | str, text: str, business connection id:
                                                          str | None = None, message_thread_id: int | None =
                                                          None, parse mode: str |
                                                          ~aiogram.client.default.Default | None =
                                                          <Default('parse mode')>, entities:
                                                          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')>, allow_paid_broadcast:
                                                          bool | None = None, message_effect_id: str | None =
                                                          None, reply_parameters:
                                                          ~aiogram.types.reply_parameters.ReplyParameters |
                                                          None = None, reply\_markup:
                                                          ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                          ~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup
                                                          ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                          | ~aiogram.types.force_reply.ForceReply | None =
                                                          None, allow sending without reply: bool | None =
                                                          None, disable_web_page_preview: bool |
                                                          ~aiogram.client.default.Default | None =
                                                          <Default('link_preview_is_disabled')>,
                                                          reply_to_message_id: int | None = None,
                                                          **extra_data: ~typing.Any)
```

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

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

text:

Text of the message to be sent, 1-4096 characters after entities parsing

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

message_thread_id: int | None

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

parse_mode: str | Default | None

Mode for parsing entities in the message text. See formatting options for more details.

entities: list[MessageEntity] | None

A JSON-serialized list of special entities that appear in message text, which can be specified instead of parse_mode

link_preview_options: LinkPreviewOptions | Default | None

Link preview generation options for the message

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

disable_web_page_preview: bool | Default | None

Disables link previews for links in this message

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_message(...)
```

Method as object

Imports:

- from aiogram.methods.send_message import SendMessage
- alias: from aiogram.methods import SendMessage

With specific bot

```
result: Message = await bot(SendMessage(...))
```

As reply into Webhook in handler

```
return SendMessage(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer()
- aiogram.types.message.Message.reply()
- aiogram.types.chat_join_request.ChatJoinRequest.answer()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply()

sendPaidMedia

Returns: Message

class aiogram.methods.send_paid_media.SendPaidMedia(*, chat_id: int | str, star_count: int, media:

list[InputPaidMediaPhoto |
InputPaidMediaVideo], business_connection_id:
str | None = None, payload: str | None = None,
caption: str | None = None, parse_mode: str |
None = None, caption_entities:
list[MessageEntity] | None = None,
show_caption_above_media: bool | None =
None, disable_notification: bool | None = None,

protect_content: bool | None = None, allow_paid_broadcast: bool | None = None, reply_parameters: ReplyParameters | None = None, reply_markup: InlineKeyboardMarkup | ReplyKeyboardMarkup | ReplyKeyboardRemove

| ForceReply | None = None, **extra_data: Any)

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

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername). If the chat is a channel, all Telegram Star proceeds from this media will be credited to the chat's balance. Otherwise, they will be credited to the bot's balance.

star_count: int

The number of Telegram Stars that must be paid to buy access to the media; 1-2500

media: list[InputPaidMediaUnion]

A JSON-serialized array describing the media to be sent; up to 10 items

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

payload: str | None

Bot-defined paid media payload, 0-128 bytes. This will not be displayed to the user, use it for your internal processes.

caption: str | None

Media caption, 0-1024 characters after entities parsing

parse_mode: str | None

Mode for parsing entities in the media caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | None

Pass True, if the caption must be shown above the message media

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

Usage

As bot method

```
result: Message = await bot.send_paid_media(...)
```

Method as object

Imports:

- from aiogram.methods.send_paid_media import SendPaidMedia
- alias: from aiogram.methods import SendPaidMedia

With specific bot

```
result: Message = await bot(SendPaidMedia(...))
```

As reply into Webhook in handler

```
return SendPaidMedia(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_paid_media()
- aiogram.types.message.Message.reply_paid_media()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_paid_media()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_paid_media()

sendPhoto

Returns: Message

```
class aiogram.methods.send_photo.SendPhoto(*, chat_id: int | str, photo: str |
                                                      ~aiogram.types.input_file.InputFile,
                                                      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:
                                                      list[~aiogram.types.message entity.MessageEntity] | None =
                                                      None, show caption above media: bool |
                                                      ~aiogram.client.default.Default | None =
                                                      <Default('show_caption_above_media')>, has_spoiler: bool
                                                      | None = None, disable\_notification: bool | None = None,
                                                      protect_content: bool | ~aiogram.client.default.Default |
                                                      None = \langle Default('protect\ content') \rangle,
                                                      allow\_paid\_broadcast: bool \mid None = None,
                                                      message_effect_id: str | None = None, reply_parameters:
                                                      ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                      None, reply_markup:
                                                      ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                      ~aiogram.types.reply keyboard markup.ReplyKeyboardMarkup
                                                      ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                      | \sim aiogram.types.force\_reply.ForceReply | None = None,
                                                      allow sending without reply: bool \mid None = None,
                                                      reply_to_message_id: int | None = None, **extra_data:
                                                      ~typing.Any)
```

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

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

photo: InputFileUnion

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

show_caption_above_media: bool | Default | None

Pass True, if the caption must be shown above the message media

has_spoiler: bool | None

Pass True if the 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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_photo(...)
```

Method as object

Imports:

- from aiogram.methods.send_photo import SendPhoto
- alias: from aiogram.methods import SendPhoto

With specific bot

```
result: Message = await bot(SendPhoto(...))
```

As reply into Webhook in handler

```
return SendPhoto(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_photo()
- aiogram.types.message.Message.reply_photo()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_photo()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_photo_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_photo()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_photo()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_photo()

sendPoll

Returns: Message

```
class aiogram.methods.send_poll.SendPoll(*, chat id: int | str, question: str, options:
                                                   list[~aiogram.types.input_poll_option.InputPollOption | str],
                                                   business connection id: str \mid None = None,
                                                   message_thread_id: int | None = None, question_parse_mode:
                                                   str | ~aiogram.client.default.Default | None =
                                                   < Default('parse mode')>, question entities:
                                                   list[~aiogram.types.message entity.MessageEntity] | None =
                                                   None, is anonymous: bool | None = None, type: str | None =
                                                   None, allows multiple answers: bool | None = None,
                                                   correct_option_id: int | None = None, explanation: str | None
                                                   = None, explanation_parse_mode: str |
                                                   ~aiogram.client.default.Default | None =
                                                   <Default('parse_mode')>, explanation_entities:
                                                   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')>, allow paid broadcast: bool |
                                                   None = None, message\_effect\_id: str | None = None,
                                                   reply parameters:
                                                   ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                   None, reply markup:
                                                   ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                   ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                   ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                   | ~aiogram.types.force_reply.ForceReply | None = None,
                                                   allow\_sending\_without\_reply: bool | None = None,
                                                   reply_to_message_id: int | None = None, **extra_data:
                                                   ~typing.Any)
```

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

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

chat_id: ChatIdUnion

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[InputPollOptionUnion]

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

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: DateTimeUnion | 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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

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

```
reply_to_message_id: int | None
```

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

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

Usage

As bot method

```
result: Message = await bot.send_poll(...)
```

Method as object

Imports:

- from aiogram.methods.send_poll import SendPoll
- alias: from aiogram.methods import SendPoll

With specific bot

```
result: Message = await bot(SendPoll(...))
```

As reply into Webhook in handler

```
return SendPoll(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_poll()
- aiogram.types.message.Message.reply_poll()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_poll()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_poll_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_poll()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_poll()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_poll()

sendVenue

```
Returns: Message
```

```
class aiogram.methods.send_venue.SendVenue(*, chat_id: int | str, latitude: float, longitude: float, title: str,
                                                      address: str, business\_connection\_id: str | None = None,
                                                      message_thread_id: int | None = None, foursquare_id: str |
                                                      None = None, foursquare\_type: str | None = None,
                                                      google_place_id: str | None = None, google_place_type: str
                                                      | None = None, disable notification: bool | None = None,
                                                      protect_content: bool | ~aiogram.client.default.Default |
                                                      None = <Default('protect content')>,
                                                      allow_paid_broadcast: bool | None = None,
                                                      message effect id: str \mid None = None, reply parameters:
                                                      \hbox{$\sim$}aiogram.types.reply\_parameters.ReplyParameters \mid None =
                                                      None, reply_markup:
                                                      ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                      ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                      ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                      | ~aiogram.types.force_reply.ForceReply | None = None,
                                                      allow_sending_without_reply: bool | None = None,
                                                      reply_to_message_id: int | None = None, **extra_data:
                                                      ~typing.Any)
```

Use this method to send information about a venue. On success, the sent aiogram.types.message.Message is returned.

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

chat_id: ChatIdUnion

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'.)

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_venue(...)
```

Method as object

Imports:

- from aiogram.methods.send_venue import SendVenue
- alias: from aiogram.methods import SendVenue

With specific bot

```
result: Message = await bot(SendVenue(...))
```

As reply into Webhook in handler

```
return SendVenue(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_venue()
- aiogram.types.message.Message.reply_venue()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_venue()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_venue_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_venue()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_venue()
- $\bullet \ aiogram.types.inaccessible_message.InaccessibleMessage.reply_venue()\\$

sendVideo

Returns: Message

```
class aiogram.methods.send_video.SendVideo(*, chat id: int | str, video: str |
                                                       ~aiogram.types.input_file.InputFile,
                                                      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, cover: str | ~aiogram.types.input file.InputFile |
                                                      None = None, start timestamp: \sim datetime.datetime
                                                      ~datetime.timedelta | int | None = None, caption: str | None
                                                      = None, parse_mode: str | ~aiogram.client.default.Default |
                                                      None = \langle Default('parse\_mode') \rangle, caption_entities:
                                                      list[~aiogram.types.message_entity.MessageEntity] | None =
                                                      None, show_caption_above_media: bool |
                                                      ~aiogram.client.default.Default | None =
                                                       <Default('show_caption_above_media')>, has_spoiler: bool
                                                      | None = None, supports\_streaming: bool | None = None,
                                                      disable_notification: bool | None = None, protect_content:
                                                      bool | ~aiogram.client.default.Default | None =
                                                       <Default('protect_content')>, allow_paid_broadcast: bool |
                                                      None = None, message effect id: str | None = None,
                                                      reply_parameters:
                                                       ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                      None, reply_markup:
                                                      ~aiogram.types.inline keyboard markup.InlineKeyboardMarkup
                                                      ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                      ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                      | \sim aiogram.types.force\_reply.ForceReply | None = None,
                                                      allow\_sending\_without\_reply: bool | None = None,
                                                      reply_to_message_id: int | None = None, **extra_data:
```

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.

~typing.Any)

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

video: InputFileUnion

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* »

cover: InputFileUnion | None

Cover for the video in the message. 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 >> *

start_timestamp: DateTimeUnion | None

Start timestamp for the video in the message

caption: str | None

Video caption (may also be used when resending videos by file_id), 0-1024 characters after entities parsing

parse_mode: str | Default | None

Mode for parsing entities in the video caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse mode

show_caption_above_media: bool | Default | None

Pass True, if the caption must be shown above the message media

has_spoiler: bool | None

Pass True if the video needs to be covered with a spoiler animation

supports_streaming: bool | None

Pass True if the uploaded video is suitable for streaming

disable notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_video(...)
```

Method as object

Imports:

- from aiogram.methods.send_video import SendVideo
- alias: from aiogram.methods import SendVideo

With specific bot

```
result: Message = await bot(SendVideo(...))
```

As reply into Webhook in handler

```
return SendVideo(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_video()
- aiogram.types.message.Message.reply_video()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_video()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_video_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_video()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_video()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_video()

sendVideoNote

```
Returns: Message
```

```
class aiogram.methods.send_video_note.SendVideoNote(*, chat_id: int | str, video_note: str |
                                                                 ~aiogram.types.input_file.InputFile,
                                                                 business\_connection\_id: str \mid None = None,
                                                                 message\_thread\_id: int \mid None = None, duration:
                                                                 int \mid None = None, length: int \mid None = None,
                                                                 thumbnail: ~aiogram.types.input file.InputFile |
                                                                 None = None, disable_notification: bool | None
                                                                 = None, protect content: bool
                                                                 ~aiogram.client.default.Default | None =
                                                                 < Default('protect content')>,
                                                                 allow paid broadcast: bool \mid None = None,
                                                                 message\_effect\_id: str \mid None = None,
                                                                 reply_parameters:
                                                                 ~aiogram.types.reply_parameters.ReplyParameters
                                                                 | None = None, reply_markup:
                                                                 ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                                 ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                                 ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                                 | ~aiogram.types.force_reply.ForceReply | None
                                                                 = None, allow_sending_without_reply: bool |
                                                                 None = None, reply to message id: int | None
                                                                 = None, **extra_data: ~typing.Any)
```

As of v.4.0, Telegram clients support rounded square MPEG4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent *aiogram.types.message* is returned.

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

video_note: InputFileUnion

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* »

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

Usage

As bot method

```
result: Message = await bot.send_video_note(...)
```

Method as object

Imports:

- from aiogram.methods.send_video_note import SendVideoNote
- alias: from aiogram.methods import SendVideoNote

With specific bot

```
result: Message = await bot(SendVideoNote(...))
```

As reply into Webhook in handler

```
return SendVideoNote(...)
```

As shortcut from received object

```
    aiogram.types.message.Message.answer_video_note()
```

- aiogram.types.message.Message.reply_video_note()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_video_note()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_video_note_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_video_note()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_video_note()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_video_note()

sendVoice

```
Returns: Message
```

```
class aiogram.methods.send_voice.SendVoice(*, chat_id: int | str, voice: str |
                                                      ~aiogram.types.input file.InputFile,
                                                     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:
                                                     list[~aiogram.types.message_entity.MessageEntity] | None =
                                                     None, duration: int \mid None = None, disable\_notification:
                                                     bool | None = None, protect_content: bool |
                                                     ~aiogram.client.default.Default | None =
                                                     <Default('protect_content')>, allow_paid_broadcast: bool |
                                                     None = None, message\_effect\_id: str | None = None,
                                                     reply_parameters:
                                                     ~aiogram.types.reply_parameters.ReplyParameters | None =
                                                     None, reply_markup:
                                                     ~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
                                                     ~aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
                                                     ~aiogram.types.reply_keyboard_remove.ReplyKeyboardRemove
                                                     | ~aiogram.types.force_reply.ForceReply | None = None,
                                                     allow\_sending\_without\_reply: bool \mid None = None,
                                                     reply_to_message_id: int | None = None, **extra_data:
                                                     ~typing.Any)
```

Use this method to send audio files, if you want Telegram clients to display the file as a playable voice message. For this to work, your audio must be in an .OGG file encoded with OPUS, or in .MP3 format, or in .M4A format (other formats may be sent as <code>aiogram.types.audio.Audio</code> or <code>aiogram.types.document.Document</code>). On success, the sent <code>aiogram.types.message</code>. <code>Message</code> is returned. Bots can currently send voice messages of up to 50 MB in size, this limit may be changed in the future.

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

voice: InputFileUnion

Audio file to send. Pass a file_id as String to send a file that exists on the Telegram servers (recommended), pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. *More information on Sending Files* »

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

message_thread_id: int | None

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

caption: str | None

Voice message caption, 0-1024 characters after entities parsing

parse_mode: str | Default | None

Mode for parsing entities in the voice message caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode

duration: int | None

Duration of the voice message in seconds

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

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

472

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: ReplyMarkupUnion | None

Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove a reply keyboard or to force a reply from the user

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

```
reply_to_message_id: int | None
```

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

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

Usage

As bot method

```
result: Message = await bot.send_voice(...)
```

Method as object

Imports:

- from aiogram.methods.send_voice import SendVoice
- alias: from aiogram.methods import SendVoice

With specific bot

```
result: Message = await bot(SendVoice(...))
```

As reply into Webhook in handler

```
return SendVoice(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_voice()
- aiogram.types.message.Message.reply_voice()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_voice()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_voice_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_voice()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_voice()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_voice()

setChatAdministratorCustomTitle

Returns: bool

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: ChatIdUnion

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

user_id: int

Unique identifier of the target user

custom_title: str

New custom title for the administrator; 0-16 characters, emoji are not allowed

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
- $\bullet \ alias: from \ aiogram.methods \ import \ SetChatAdministratorCustomTitle\\$

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

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

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 | None = None,

**extra data: Any)

Use this method to change the bot's menu button in a private chat, or the default menu button. Returns True on success.

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

chat_id: int | None

Unique identifier for the target private chat. If not specified, default bot's menu button will be changed

menu_button: MenuButtonUnion | None

A JSON-serialized object for the bot's new menu button. Defaults to aiogram.types.menu_button_default.MenuButtonDefault

Usage

As bot method

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

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: ChatIdUnion

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

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

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: ChatIdUnion

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

sticker_set_name: str

Name of the sticker set to be set as the group sticker set

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

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 |
ReactionTypePaid] | None = None,
is_big: bool | None = None,
**extra_data: Any)

Use this method to change the chosen reactions on a message. Service messages of some types can't be reacted to. Automatically forwarded messages from a channel to its discussion group have the same available reactions as messages in the channel. Bots can't use paid reactions. Returns True on success.

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

```
chat_id: int | str
```

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

message_id: int

Identifier of the target message. If the message belongs to a media group, the reaction is set to the first non-deleted message in the group instead.

```
reaction: list[ReactionTypeEmoji | ReactionTypeCustomEmoji | ReactionTypePaid] |
None
```

A JSON-serialized list of reaction types to set on the message. Currently, as non-premium users, bots can set up to one reaction per message. A custom emoji reaction can be used if it is either already present on the message or explicitly allowed by chat administrators. Paid reactions can't be used by bots.

is_big: bool | None

Pass True to set the reaction with a big animation

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.set_message_reaction import SetMessageReaction
- alias: from aiogram.methods import SetMessageReaction

With specific bot

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

As reply into Webhook in handler

```
return SetMessageReaction(...)
```

As shortcut from received object

• aiogram.types.message.Message.react()

setMyCommands

Returns: bool

class aiogram.methods.set_my_commands.SetMyCommands(*, commands: list[BotCommand], scope:

BotCommandScopeDefault |

BotCommandScopeAllPrivateChats |

BotCommandScopeAllGroupChats |

BotCommandScopeAllChatAdministrators |

BotCommandScopeChat |

BotCommandScopeChatAdministrators |

 $BotCommandScopeChatMember \mid None = None,$

 $language_code : str \mid None = None,$

**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: BotCommandScopeUnion | None

A JSON-serialized object, describing scope of users for which the commands are relevant. Defaults to aiogram.types.bot_command_scope_default.BotCommandScopeDefault.

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

 ${\bf class} \ {\bf aiogram.methods.set_my_default_administrator_rights. \\ {\bf SetMyDefaultAdministratorRights} (*, {\bf class}) \\ {\bf class} \ {$

rights: ChatAdministrator-Rights None None, for_channels: boolNone None, **extra_data: Any)

Use this method to change the default administrator rights requested by the bot when it's added as an administrator to groups or channels. These rights will be suggested to users, but they are free to modify the list before adding the bot. Returns True on success.

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

rights: ChatAdministratorRights | None

A JSON-serialized object describing new default administrator rights. If not specified, the default administrator rights will be cleared.

for_channels: bool | None

Pass True to change the default administrator rights of the bot in channels. Otherwise, the default administrator rights of the bot for groups and supergroups will be changed.

As bot method

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

Method as object

Imports:

- from aiogram.methods.set_my_default_administrator_rights import SetMyDefaultAdministratorRights
- alias: from aiogram.methods import SetMyDefaultAdministratorRights

With specific bot

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

As reply into Webhook in handler

```
return SetMyDefaultAdministratorRights(...)
```

setMyDescription

Returns: bool

Use this method to change the bot's description, which is shown in the chat with the bot if the chat is empty. Returns True on success.

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

description: str | None

New bot description; 0-512 characters. Pass an empty string to remove the dedicated description for the given language.

language_code: str | None

A two-letter ISO 639-1 language code. If empty, the description will be applied to all users for whose language there is no dedicated description.

As bot method

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

Method as object

Imports:

- from aiogram.methods.set_my_description import SetMyDescription
- alias: from aiogram.methods import SetMyDescription

With specific bot

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

As reply into Webhook in handler

```
return SetMyDescription(...)
```

setMyName

Returns: bool

Use this method to change the bot's name. Returns True on success.

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

name: str | None

New bot name; 0-64 characters. Pass an empty string to remove the dedicated name for the given language.

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(...)
```

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

```
class aiogram.methods.set_my_short_description.SetMyShortDescription(*, short\_description: str \mid None = None, language\_code: str \mid None = None, **extra_data: Any)
```

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.

```
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(...)
```

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(...)
```

setUserEmojiStatus

```
Returns: bool
```

Changes the emoji status for a given user that previously allowed the bot to manage their emoji status via the Mini App method requestEmojiStatusAccess. Returns True on success.

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

```
user_id: int
```

Unique identifier of the target user

```
emoji_status_custom_emoji_id: str | None
```

Custom emoji identifier of the emoji status to set. Pass an empty string to remove the status.

```
emoji_status_expiration_date: DateTimeUnion | None
```

Expiration date of the emoji status, if any

Usage

488

As bot method

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

Imports:

- from aiogram.methods.set_user_emoji_status import SetUserEmojiStatus
- alias: from aiogram.methods import SetUserEmojiStatus

With specific bot

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

As reply into Webhook in handler

```
return SetUserEmojiStatus(...)
```

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: ChatIdUnion

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

only_if_banned: bool | None

Do nothing if the user is not banned

Usage

As bot method

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

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

```
sender_chat_id: int
```

Unique identifier of the target sender chat

Usage

As bot method

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

Imports:

- from aiogram.methods.unban_chat_sender_chat import UnbanChatSenderChat
- alias: from aiogram.methods import UnbanChatSenderChat

With specific bot

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

As reply into Webhook in handler

```
return UnbanChatSenderChat(...)
```

As shortcut from received object

• aiogram.types.chat.Chat.unban_sender_chat()

unhideGeneralForumTopic

Returns: bool

Use this method to unhide the 'General' topic in a forum supergroup chat. The bot must be an administrator in the chat for this to work and must have the *can manage topics* administrator rights. Returns True on success.

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

chat_id: ChatIdUnion

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

Usage

As bot method

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

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.unpin_all_chat_messages import UnpinAllChatMessages
- alias: from aiogram.methods import UnpinAllChatMessages

With specific bot

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

As reply into Webhook in handler

```
return UnpinAllChatMessages(...)
```

As shortcut from received object

• aiogram.types.chat.Chat.unpin_all_messages()

unpinAllForumTopicMessages

Returns: bool

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: ChatIdUnion

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

Usage

As bot method

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

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

class aiogram.methods.unpin_all_general_forum_topic_messages.UnpinAllGeneralForumTopicMessages(*,

```
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)

Usage

As bot method

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

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: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be unpinned

```
message_id: int | None
```

Identifier of the message to unpin. Required if *business_connection_id* is specified. If not specified, the most recent pinned message (by sending date) will be unpinned.

As bot method

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

Method as object

Imports:

- from aiogram.methods.unpin_chat_message import UnpinChatMessage
- alias: from aiogram.methods import UnpinChatMessage

With specific bot

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

As reply into Webhook in handler

```
return UnpinChatMessage(...)
```

As shortcut from received object

- aiogram.types.chat.Chat.unpin_message()
- aiogram.types.message.Message.unpin()

Updating messages

deleteMessage

Returns: bool

Use this method to delete a message, including service messages, with the following limitations:

- A message can only be deleted if it was sent less than 48 hours ago.
- Service messages about a supergroup, channel, or forum topic creation can't be deleted.
- A dice message in a private chat can only be deleted if it was sent more than 24 hours ago.
- Bots can delete outgoing messages in private chats, groups, and supergroups.
- Bots can delete incoming messages in private chats.
- Bots granted *can_post_messages* permissions can delete outgoing messages in channels.
- If the bot is an administrator of a group, it can delete any message there.

• If the bot has *can_delete_messages* permission in a supergroup or a channel, it can delete any message there.

Returns True on success.

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

message_id: int

Identifier of the message to delete

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)

```
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(...)
```

~typing.Any)

editMessageCaption

```
Returns: Union[Message, bool]
```

```
class aiogram.methods.edit_message_caption.EditMessageCaption(*, business_connection_id: str |
                                                                               None = None, chat\_id: int | str |
                                                                               None = None, message\_id: int |
                                                                               None = None, inline\_message\_id:
                                                                               str \mid None = None, caption: str \mid
                                                                               None = None, parse mode: str
                                                                               ~aiogram.client.default.Default |
                                                                               None = \langle Default('parse\_mode') \rangle,
                                                                               caption_entities:
                                                                               list[~aiogram.types.message entity.MessageEntity]
                                                                               | None = None,
                                                                               show_caption_above_media: bool |
                                                                               ~aiogram.client.default.Default |
                                                                               None = < De-
                                                                               fault('show_caption_above_media')>,
                                                                               reply markup:
                                                                               ~aiogram.types.inline_keyboard_markup.InlineKeybo
                                                                               | None = None, **extra_data:
```

Use this method to edit captions of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours from the time they were sent.

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

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

chat_id: ChatIdUnion | 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

parse_mode: str | Default | None

Mode for parsing entities in the message caption. See formatting options for more details.

caption_entities: list[MessageEntity] | None

A JSON-serialized list of special entities that appear in the caption, which can be specified instead of parse_mode

show_caption_above_media: bool | Default | None

Pass True, if the caption must be shown above the message media. Supported only for animation, photo and video messages.

```
reply_markup: InlineKeyboardMarkup | None A JSON-serialized object for an inline keyboard.
```

As bot method

```
result: Union[Message, bool] = await bot.edit_message_caption(...)
```

Method as object

Imports:

- $\bullet \ \, from \ \, aiogram.methods.edit_message_caption \ \, import \ \, EditMessageCaption$
- 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, business_connection_id: $str \mid None = None,$ chat id: int | str | None = None. message id: int None = None. inline_message_id: $str \mid None = None$, live_period: int | None = None, horizontal_accuracy: $float \mid None = None,$ heading: int | None = None, proximity alert radius: int | None = None,reply markup: InlineKeyboardMarkup | None = None,**extra_data: Any)

Use this method to edit live location messages. A location can be edited until its *live_period* expires or editing is explicitly disabled by a call to <code>aiogram.methods.stop_message_live_location</code>. StopMessageLiveLocation. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned.

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

latitude: float

Latitude of new location

longitude: float

Longitude of new location

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

chat_id: ChatIdUnion | 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

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,

business_connection_id: str | None =
None, chat_id: int | str | None = None,
message_id: int | None = None,
inline_message_id: str | None = None,
reply_markup: InlineKeyboardMarkup |
None = None, **extra_data: Any)

Use this method to edit animation, audio, document, photo, or video messages, or to add media to text messages. If a message is part of a message album, then it can be edited only to an audio for audio albums, only to a document for document albums and to a photo or a video otherwise. When an inline message is edited, a new file can't be uploaded; use a previously uploaded file via its file_id or specify a URL. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within **48 hours** from the time they were sent.

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

media: InputMediaUnion

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

```
business_connection_id: str | None
```

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
chat_id: ChatIdUnion | 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

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for a new inline keyboard.

Usage

As bot method

```
result: Union[Message, bool] = await bot.edit_message_media(...)
```

Method as object

Imports:

- from aiogram.methods.edit_message_media import EditMessageMedia
- alias: from aiogram.methods import EditMessageMedia

With specific bot

```
result: Union[Message, bool] = await bot(EditMessageMedia(...))
```

As reply into Webhook in handler

```
return EditMessageMedia(...)
```

As shortcut from received object

aiogram.types.message.Message.edit_media()

editMessageReplyMarkup

Returns: Union[Message, bool]

class aiogram.methods.edit_message_reply_markup.EditMessageReplyMarkup(*, busi-

ness_connection_id: str | None = None, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, **extra_data: Any)

Use this method to edit only the reply markup of messages. On success, if the edited message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise <code>True</code> is returned. Note that business messages that were not sent by the bot and do not contain an inline keyboard can only be edited within <code>48 hours</code> from the time they were sent.

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

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

chat_id: ChatIdUnion | 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

reply_markup: InlineKeyboardMarkup | None

A JSON-serialized object for an inline keyboard.

Usage

As bot method

```
result: Union[Message, bool] = await bot.edit_message_reply_markup(...)
```

Method as object

Imports:

- from aiogram.methods.edit_message_reply_markup import EditMessageReplyMarkup
- alias: from aiogram.methods import EditMessageReplyMarkup

With specific bot

```
result: Union[Message, bool] = await bot(EditMessageReplyMarkup(...))
```

As reply into Webhook in handler

```
return EditMessageReplyMarkup(...)
```

As shortcut from received object

- aiogram.types.message.Message.edit_reply_markup()
- aiogram.types.message.Message.delete_reply_markup()

editMessageText

Returns: Union[Message, bool]

```
class aiogram.methods.edit_message_text.EditMessageText(*, text: str, business connection id: str|
                                                                      None = None, chat\_id: int | str | None =
                                                                     None, message id: int | None = None,
                                                                     inline_message_id: str | None = None,
                                                                     parse mode: str |
                                                                      ~aiogram.client.default.Default
                                                                      |None = \langle Default('parse\ mode') \rangle, entities:
                                                                      list[~aiogram.types.message_entity.MessageEntity]
                                                                      | None = None, link_preview_options:
                                                                      ~aiogram.types.link_preview_options.LinkPreviewOptions
                                                                     | ~aiogram.client.default.Default | None =
                                                                      <Default('link_preview')>, reply_markup:
                                                                      ~aiogram.types.inline_keyboard_markup.InlineKeyboardMark
                                                                      | None = None,
                                                                      disable_web_page_preview: bool |
                                                                      ~aiogram.client.default.Default | None =
                                                                      < Default('link_preview_is_disabled')>,
                                                                      **extra data: ~typing.Any)
     Use this method to edit text and game messages. On success, if the edited message is not an inline message,
     the edited aiogram.types.message.Message is returned, otherwise True is returned. Note that business
     messages that were not sent by the bot and do not contain an inline keyboard can only be edited within 48 hours
     from the time they were sent.
     Source: https://core.telegram.org/bots/api#editmessagetext
     text: str
           New text of the message, 1-4096 characters after entities parsing
     business_connection_id: str | None
           Unique identifier of the business connection on behalf of which the message to be edited was sent
     chat_id: ChatIdUnion | 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.
     entities: list[MessageEntity] | None
           A JSON-serialized list of special entities that appear in message text, which can be specified instead of
           parse_mode
```

link_preview_options: LinkPreviewOptions | Default | None

Link preview generation options for the message

reply_markup: InlineKeyboardMarkup | None

A JSON-serialized object for an inline keyboard.

disable_web_page_preview: bool | Default | None

Disables link previews for links in this message

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

As bot method

```
result: Union[Message, bool] = await bot.edit_message_text(...)
```

Method as object

Imports:

- from aiogram.methods.edit_message_text import EditMessageText
- alias: from aiogram.methods import EditMessageText

With specific bot

```
result: Union[Message, bool] = await bot(EditMessageText(...))
```

As reply into Webhook in handler

```
return EditMessageText(...)
```

As shortcut from received object

• aiogram.types.message.Message.edit_text()

stopMessageLiveLocation

```
Returns: Union[Message, bool]
```

class aiogram.methods.stop_message_live_location.StopMessageLiveLocation(*, busi-

ness_connection_id: str | None = None, chat_id: int | str | None = None, message_id: int | None = None, inline_message_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, **extra_data: Any)

Use this method to stop updating a live location message before *live_period* expires. On success, if the message is not an inline message, the edited <code>aiogram.types.message.Message</code> is returned, otherwise <code>True</code> is returned.

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

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

chat_id: ChatIdUnion | 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

```
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
```

```
class aiogram.methods.stop_poll.StopPoll(*, chat_id: int | str, message_id: int, business_connection_id: str | None = None, reply_markup: InlineKeyboardMarkup | None = None, **extra_data: Any)
```

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

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

chat_id: ChatIdUnion

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

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message to be edited was sent

```
reply_markup: InlineKeyboardMarkup | None
```

A JSON-serialized object for a new message inline keyboard.

Usage

As bot method

```
result: Poll = await bot.stop_poll(...)
```

Method as object

Imports:

- from aiogram.methods.stop_poll import StopPoll
- alias: from aiogram.methods import StopPoll

With specific bot

```
result: Poll = await bot(StopPoll(...))
```

As reply into Webhook in handler

```
return StopPoll(...)
```

Inline mode

answerInlineQuery

Returns: bool

class aiogram.methods.answer_inline_query.AnswerInlineQuery(*, inline_query_id: str, results:

list[InlineQueryResultCachedAudio |

InlineQueryResultCachedDocument |

InlineQueryResultCachedGif |

InlineQueryResultCachedMpeg4Gif |

InlineQueryResultCachedPhoto |

InlineQueryResultCachedSticker |

 $In line Query Result Cached Video \mid$

InlineQueryResultCachedVoice |

InlineQueryResultArticle |

InlineQueryResultAudio |

InlineQueryResultContact |

InlineQueryResultGame |

InlineQueryResultDocument |

InlineQueryResultGif |

InlineQueryResultLocation |

InlineQueryResultMpeg4Gif |

InlineQueryResultPhoto |

InlineQueryResultVenue |

InlineQueryResultVideo |

InlineQueryResultVoice], cache_time:

int | None = None, is_personal: bool |

 $None = None, next_offset: str \mid None$

= *None*, button:

InlineQueryResultsButton | None =

None, switch_pm_parameter: str |

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[InlineQueryResultUnion]

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.

next_offset: str | None

Pass the offset that a client should send in the next query with the same text to receive more results. Pass an empty string if there are no more results or if you don't support pagination. Offset length can't exceed 64 bytes.

button: InlineQueryResultsButton | None

A JSON-serialized object describing a button to be shown above inline query results

switch_pm_parameter: str | None

Deep-linking parameter for the /start message sent to the bot when user presses the switch button. 1-64 characters, only A-Z, a-z, 0-9, _ and - are allowed.

Deprecated since version API:6.7: https://core.telegram.org/bots/api-changelog#april-21-2023

switch_pm_text: str | None

If passed, clients will display a button with specified text that switches the user to a private chat with the bot and sends the bot a start message with the parameter *switch_pm_parameter*

Deprecated since version API:6.7: https://core.telegram.org/bots/api-changelog#april-21-2023

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.answer_inline_query import AnswerInlineQuery
- alias: from aiogram.methods import AnswerInlineQuery

With specific bot

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

As reply into Webhook in handler

```
return AnswerInlineQuery(...)
```

As shortcut from received object

• aiogram.types.inline_query.InlineQuery.answer()

answerWebAppQuery

Returns: SentWebAppMessage

class aiogram.methods.answer_web_app_query.AnswerWebAppQuery(*, web_app_query_id: str, result:

InlineQueryResultCachedAudio | InlineQueryResultCachedDocument | InlineQueryResultCachedGif | InlineQueryResultCachedMpeg4Gif | InlineQueryResultCachedPhoto | InlineQueryResultCachedSticker | InlineQueryResultCachedVideo | InlineQueryResultCachedVoice | InlineQueryResultArticle | InlineOueryResultAudio | InlineQueryResultContact | InlineQueryResultGame | InlineQueryResultDocument | InlineQueryResultGif | InlineQueryResultLocation | InlineQueryResultMpeg4Gif | InlineQueryResultPhoto | InlineQueryResultVenue | InlineQueryResultVideo | InlineQueryResultVoice, **extra data: Any)

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

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

web_app_query_id: str

Unique identifier for the query to be answered

result: InlineQueryResultUnion

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(...)
```

savePreparedInlineMessage

Returns: PreparedInlineMessage

 $\textbf{class} \ \, \textbf{aiogram.methods.save_prepared_inline_message}. \textbf{SavePreparedInlineMessage}(*, \textit{user_id: int}, \\$

result: Inline-

QueryResult-

CachedAudio

InlineQueryRe-

sultCachedDoc-

ument I

InlineQueryRe-

sultCachedGif |

InlineQueryRe-

sultCachedM-

peg4Gif |

InlineQueryRe-

sultCached-

Photo |

InlineQueryRe-

sultCached-

Sticker

InlineQueryRe-

sultCached-

Video

InlineQueryRe-

sultCached-

Voice

InlineQueryRe-

sultArticle |

InlineQueryRe-

sultAudio |

InlineQueryRe-

sultContact |

InlineQueryRe-

sultGame |

InlineQueryRe-

sultDocument |

InlineQueryRe-

sultGif |

InlineQueryRe-

sultLocation |

InlineQueryRe-

sultMpeg4Gif |

InlineQueryRe-

sultPhoto

InlineQueryRe-

sultVenue |

InlineQueryRe-

sultVideo

Suit viuco

InlineQueryRe-

sultVoice,

al-

low_user_chats:

 $bool \mid None =$

None, al-

low_bot_chats:

bool | None =

None, al-

low_group_chats:

Stores a message that can be sent by a user of a Mini App. Returns a aiogram.types. prepared_inline_message.PreparedInlineMessage object.

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

user_id: int

Unique identifier of the target user that can use the prepared message

result: InlineQueryResultUnion

A JSON-serialized object describing the message to be sent

allow_user_chats: bool | None

Pass True if the message can be sent to private chats with users

allow_bot_chats: bool | None

Pass True if the message can be sent to private chats with bots

allow_group_chats: bool | None

Pass True if the message can be sent to group and supergroup chats

allow_channel_chats: bool | None

Pass True if the message can be sent to channel chats

Usage

As bot method

```
result: PreparedInlineMessage = await bot.save_prepared_inline_message(...)
```

Method as object

Imports:

- from aiogram.methods.save_prepared_inline_message import SavePreparedInlineMessage
- alias: from aiogram.methods import SavePreparedInlineMessage

With specific bot

```
result: PreparedInlineMessage = await bot(SavePreparedInlineMessage(...))
```

As reply into Webhook in handler

```
return SavePreparedInlineMessage(...)
```

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

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:

- $\bullet \ \, from \ \, aiogram.methods.get_game_high_scores \ \, import \ \, GetGameHighScores$
- 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,
```

 $business_connection_id: str \mid None = None,$

 $message_thread_id: int \mid None = None, disable_notification:$

 $bool \mid None = None, protect_content: bool \mid$

 ${\sim} aiogram.client.default.Default \mid None =$

None = None, message_effect_id: str | None = None,

reply_parameters:

~aiogram.types.reply_parameters.ReplyParameters | None =

None, reply_markup:

~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup | None = None, allow_sending_without_reply: bool | None =

 $None, reply_to_message_id: int \mid None = None, **extra_data:$

~typing.Any)

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

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

chat_id: int

Unique identifier for the target chat

game_short_name: str

Short name of the game, serves as the unique identifier for the game. Set up your games via @BotFather.

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the message will be sent

message_thread_id: int | None

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

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: InlineKeyboardMarkup | None

A JSON-serialized object for an inline keyboard. If empty, one 'Play game_title' button will be shown. If not empty, the first button must launch the game.

```
allow_sending_without_reply: bool | None
```

Pass True if the message should be sent even if the specified replied-to message is not found

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

```
reply_to_message_id: int | None
```

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

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

Usage

As bot method

```
result: Message = await bot.send_game(...)
```

Method as object

Imports:

- $\hbox{\color{red} \bullet from aiogram.methods.send_game import SendGame}\\$
- alias: from aiogram.methods import SendGame

With specific bot

```
result: Message = await bot(SendGame(...))
```

As reply into Webhook in handler

```
return SendGame(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_game()
- aiogram.types.message.Message.reply_game()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_game()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_game_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_game()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_game()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_game()

setGameScore

Returns: Union[Message, bool]

Use this method to set the score of the specified user in a game message. On success, if the message is not an inline message, the <code>aiogram.types.message.Message</code> is returned, otherwise True is returned. Returns an error, if the new score is not greater than the user's current score in the chat and <code>force</code> is <code>False</code>.

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

user_id: int

User identifier

score: int

New score, must be non-negative

```
force: bool | None
```

Pass True if the high score is allowed to decrease. This can be useful when fixing mistakes or banning cheaters

```
disable_edit_message: bool | None
```

Pass True if the game message should not be automatically edited to include the current scoreboard

```
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.

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)

```
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.

As bot method

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

Method as object

Imports:

- from aiogram.methods.answer_shipping_query import AnswerShippingQuery
- alias: from aiogram.methods import AnswerShippingQuery

With specific bot

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

As reply into Webhook in handler

```
return AnswerShippingQuery(...)
```

As shortcut from received object

• aiogram.types.shipping_query.ShippingQuery.answer()

createInvoiceLink

Returns: str

class aiogram.methods.create_invoice_link.CreateInvoiceLink(*, title: str, description: str, payload:

str, currency: str, prices: list/LabeledPrice], business_connection_id: str | None = None, provider token: str | None = None, subscription period: int | None = None, max_tip_amount: int | None = *None*, *suggested_tip_amounts*: $list[int] \mid None = None,$ provider data: $str \mid None = None$, photo $url: str \mid None = None$, $photo_size: int \mid None = None,$ $photo_width: int \mid None = None,$ $photo_height: int \mid None = None,$ $need_name: bool \mid None = None,$ 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 it for your internal processes.

currency: str

Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

prices: list[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

business_connection_id: str | None

Unique identifier of the business connection on behalf of which the link will be created. For payments in Telegram Stars only.

provider_token: str | None

Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

subscription_period: int | None

The number of seconds the subscription will be active for before the next payment. The currency must be set to 'XTR' (Telegram Stars) if the parameter is used. Currently, it must always be 2592000 (30 days) if specified. Any number of subscriptions can be active for a given bot at the same time, including multiple concurrent subscriptions from the same user. Subscription price must no exceed 2500 Telegram Stars.

max_tip_amount: int | None

The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.

suggested_tip_amounts: list[int] | None

A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.

provider_data: str | None

JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.

photo_url: str | None

URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service.

photo_size: int | None

Photo size in bytes

photo_width: int | None

Photo width

photo_height: int | None

Photo height

need_name: bool | None

Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.

need_phone_number: bool | None

Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.

need_email: bool | None

Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.

need_shipping_address: bool | None

Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

send_phone_number_to_provider: bool | None

Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars.

send_email_to_provider: bool | None

Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars.

is flexible: bool | None

Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

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(...)
```

editUserStarSubscription

Returns: bool

Allows the bot to cancel or re-enable extension of a subscription paid in Telegram Stars. Returns True on success.

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

```
user_id: int
```

Identifier of the user whose subscription will be edited

telegram_payment_charge_id: str

Telegram payment identifier for the subscription

is_canceled: bool

Pass True to cancel extension of the user subscription; the subscription must be active up to the end of the current subscription period. Pass False to allow the user to re-enable a subscription that was previously canceled by the bot.

As bot method

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

Method as object

Imports:

- from aiogram.methods.edit_user_star_subscription import EditUserStarSubscription
- alias: from aiogram.methods import EditUserStarSubscription

With specific bot

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

As reply into Webhook in handler

```
return EditUserStarSubscription(...)
```

getStarTransactions

Returns: StarTransactions

Returns the bot's Telegram Star transactions in chronological order. On success, returns a aiogram.types.star_transactions.StarTransactions object.

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

offset: int | None

Number of transactions to skip in the response

limit: int | None

The maximum number of transactions to be retrieved. Values between 1-100 are accepted. Defaults to 100.

As bot method

```
result: StarTransactions = await bot.get_star_transactions(...)
```

Method as object

Imports:

- $\bullet \ \, from \ \, aiogram.methods.get_star_transactions \ \, import \ \, GetStarTransactions \\$
- alias: from aiogram.methods import GetStarTransactions

With specific bot

```
result: StarTransactions = await bot(GetStarTransactions(...))
```

refundStarPayment

```
Returns: bool
```

Refunds a successful payment in Telegram Stars. Returns True on success.

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

```
user_id: int
```

Identifier of the user whose payment will be refunded

```
telegram_payment_charge_id: str
```

Telegram payment identifier

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.refund_star_payment import RefundStarPayment
- alias: from aiogram.methods import RefundStarPayment

With specific bot

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

As reply into Webhook in handler

```
return RefundStarPayment(...)
```

sendInvoice

Returns: Message

class aiogram.methods.send_invoice.**SendInvoice**(*, chat_id: int | str, title: str, description: str, payload:

str, currency: str, prices:

list[~aiogram.types.labeled_price.LabeledPrice], *message_thread_id: int* | *None* = *None*, *provider_token:* str | None = None, max_tip_amount: int | None = *None, suggested tip amounts: list[int]* $\mid None = None$, start_parameter: str | None = None, provider_data: str | None = None, photo url: str | None = None,photo_size: int | None = None, photo_width: int | None = None, photo height: int | None = None, need name: $bool \mid None = None, need_phone_number: bool \mid None$ $= None, need_email: bool | None = None,$ $need_shipping_address: bool \mid None = None,$ *send_phone_number_to_provider: bool* | *None* = *None*, $send_email_to_provider: bool \mid None = None,$ *is_flexible: bool* | *None* = *None, disable_notification:* bool | None = None, protect_content: bool | ~aiogram.client.default.Default | None = <Default('protect_content')>, allow_paid_broadcast: bool | None = None, message_effect_id: str | None = *None, reply_parameters:* ~aiogram.types.reply_parameters.ReplyParameters |

 $None = None, reply_markup$:

~aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup | None = None, allow sending without reply: bool |*None* = *None*, *reply_to_message_id*: *int* | *None* = *None*, **extra_data: ~typing.Any)

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

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

chat_id: ChatIdUnion

Unique identifier for the target chat or username of the target channel (in the format @channelusername)

title: str

Product name, 1-32 characters

description: str

Product description, 1-255 characters

payload: str

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

currency: str

Three-letter ISO 4217 currency code, see more on currencies. Pass 'XTR' for payments in Telegram Stars.

prices: list[LabeledPrice]

Price breakdown, a JSON-serialized list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.). Must contain exactly one item for payments in Telegram Stars.

message_thread_id: int | None

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

provider_token: str | None

Payment provider token, obtained via @BotFather. Pass an empty string for payments in Telegram Stars.

max_tip_amount: int | None

The maximum accepted amount for tips in the *smallest units* of the currency (integer, **not** float/double). For example, for a maximum tip of US\$ 1.45 pass max_tip_amount = 145. See the *exp* parameter in currencies.json, it shows the number of digits past the decimal point for each currency (2 for the majority of currencies). Defaults to 0. Not supported for payments in Telegram Stars.

suggested_tip_amounts: list[int] | None

A JSON-serialized array of suggested amounts of tips in the *smallest units* of the currency (integer, **not** float/double). At most 4 suggested tip amounts can be specified. The suggested tip amounts must be positive, passed in a strictly increased order and must not exceed *max_tip_amount*.

start_parameter: str | None

Unique deep-linking parameter. If left empty, **forwarded copies** of the sent message will have a *Pay* button, allowing multiple users to pay directly from the forwarded message, using the same invoice. If non-empty, forwarded copies of the sent message will have a *URL* button with a deep link to the bot (instead of a *Pay* button), with the value used as the start parameter

provider_data: str | None

JSON-serialized data about the invoice, which will be shared with the payment provider. A detailed description of required fields should be provided by the payment provider.

photo_url: str | None

URL of the product photo for the invoice. Can be a photo of the goods or a marketing image for a service. People like it better when they see what they are paying for.

photo_size: int | None

Photo size in bytes

photo_width: int | None

Photo width

photo_height: int | None

Photo height

need_name: bool | None

Pass True if you require the user's full name to complete the order. Ignored for payments in Telegram Stars.

need_phone_number: bool | None

Pass True if you require the user's phone number to complete the order. Ignored for payments in Telegram Stars.

need_email: bool | None

Pass True if you require the user's email address to complete the order. Ignored for payments in Telegram Stars.

need_shipping_address: bool | None

Pass True if you require the user's shipping address to complete the order. Ignored for payments in Telegram Stars.

send_phone_number_to_provider: bool | None

Pass True if the user's phone number should be sent to the provider. Ignored for payments in Telegram Stars

send_email_to_provider: bool | None

Pass True if the user's email address should be sent to the provider. Ignored for payments in Telegram Stars

is_flexible: bool | None

Pass True if the final price depends on the shipping method. Ignored for payments in Telegram Stars.

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | Default | None

Protects the contents of the sent message from forwarding and saving

allow_paid_broadcast: bool | None

Pass True to allow up to 1000 messages per second, ignoring broadcasting limits for a fee of 0.1 Telegram Stars per message. The relevant Stars will be withdrawn from the bot's balance

message_effect_id: str | None

Unique identifier of the message effect to be added to the message; for private chats only

reply_parameters: ReplyParameters | None

Description of the message to reply to

reply_markup: InlineKeyboardMarkup | None

A JSON-serialized object for an inline keyboard. If empty, one 'Pay total price' button will be shown. If not empty, the first button must be a Pay button.

allow_sending_without_reply: bool | None

Pass True if the message should be sent even if the specified replied-to message is not found

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

reply_to_message_id: int | None

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

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

As bot method

```
result: Message = await bot.send_invoice(...)
```

Method as object

Imports:

- from aiogram.methods.send_invoice import SendInvoice
- alias: from aiogram.methods import SendInvoice

With specific bot

```
result: Message = await bot(SendInvoice(...))
```

As reply into Webhook in handler

```
return SendInvoice(...)
```

As shortcut from received object

- aiogram.types.message.Message.answer_invoice()
- aiogram.types.message.Message.reply_invoice()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_invoice()
- aiogram.types.chat_join_request.ChatJoinRequest.answer_invoice_pm()
- aiogram.types.chat_member_updated.ChatMemberUpdated.answer_invoice()
- aiogram.types.inaccessible_message.InaccessibleMessage.answer_invoice()
- aiogram.types.inaccessible_message.InaccessibleMessage.reply_invoice()

Getting updates

deleteWebhook

Returns: bool

Use this method to remove we hook integration if you decide to switch back to aiogram.methods. $get_updates.$ Returns True on success.

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

drop_pending_updates: bool | None

Pass True to drop all pending updates

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, limit: 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.

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

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

```
class aiogram.methods.set_webhook.(*, url: str, certificate: InputFile | None = None, ip_address: str | None = None, max_connections: int | None = None, allowed_updates: list[str] | None = None, drop_pending_updates: bool | None = None, secret_token: str | None = None, **extra_data: Any)
```

Use this method to specify a URL and receive incoming updates via an outgoing webhook. Whenever there is an update for the bot, we will send an HTTPS POST request to the specified URL, containing a JSON-serialized <code>aiogram.types.update.Update</code>. In case of an unsuccessful request (a request with response HTTP status code different from 2XY), we will repeat the request and 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.

allowed_updates: list[str] | None

A JSON-serialized list of the update types you want your bot to receive. For example, specify ["message", "edited_channel_post", "callback_query"] to only receive updates of these types. See aiogram. types.update.Update for a complete list of available update types. Specify an empty list to receive all update types except chat_member, message_reaction, and message_reaction_count (default). If not specified, the previous setting will be used.

drop_pending_updates: bool | None

Pass True to drop all pending updates

secret_token: str | None

A secret token to be sent in a header 'X-Telegram-Bot-Api-Secret-Token' in every webhook request, 1-256 characters. Only characters A-Z, a-z, 0-9, _ and - are allowed. The header is useful to ensure that the request comes from a webhook set by you.

Usage

As bot method

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

Method as object

Imports:

- from aiogram.methods.set_webhook import SetWebhook
- alias: from aiogram.methods import SetWebhook

With specific bot

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

As reply into Webhook in handler

return SetWebhook(...)

Telegram Passport

setPassportDataErrors

Returns: bool

class aiogram.methods.set_passport_data_errors.SetPassportDataErrors(*, user_id: int, errors:

list[PassportElementErrorDataField

| PassportElementError-

FrontSide

PassportElementErrorRe-

verseSide |

PassportElementError-

Selfie |

PassportElementErrorFile

| PassportElementError-

Files

PassportElementError-

TranslationFile |

PassportElementError-

TranslationFiles |

PassportElementErrorUn-

specified], **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

errors: list[PassportElementErrorUnion]

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

ANIMATION = 'animation'

DOCUMENT = 'document'

AUDIO = 'audio'

```
class aiogram.enums.chat_member_status.ChatMemberStatus(value, names=None, *, module=None,
                                                              qualname=None, type=None, start=1,
                                                              boundary=None)
     This object represents chat member status.
     Source: https://core.telegram.org/bots/api#chatmember
     CREATOR = 'creator'
     ADMINISTRATOR = 'administrator'
     MEMBER = 'member'
     RESTRICTED = 'restricted'
     LEFT = 'left'
     KICKED = 'kicked'
ChatType
class aiogram.enums.chat_type.ChatType(value, names=None, *, module=None, qualname=None,
                                           type=None, start=1, boundary=None)
     This object represents a chat type
     Source: https://core.telegram.org/bots/api#chat
     SENDER = 'sender'
     PRIVATE = 'private'
     GROUP = 'group'
     SUPERGROUP = 'supergroup'
     CHANNEL = 'channel'
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'
```

```
PAID_MEDIA = 'paid_media'
PHOTO = 'photo'
STICKER = 'sticker'
STORY = 'story'
VIDEO = 'video'
VIDEO_NOTE = 'video_note'
VOICE = 'voice'
CONTACT = 'contact'
DICE = 'dice'
GAME = 'game'
POLL = 'poll'
VENUE = 'venue'
LOCATION = 'location'
NEW_CHAT_MEMBERS = 'new_chat_members'
LEFT_CHAT_MEMBER = 'left_chat_member'
NEW_CHAT_TITLE = 'new_chat_title'
NEW_CHAT_PHOTO = 'new_chat_photo'
DELETE_CHAT_PHOTO = 'delete_chat_photo'
GROUP_CHAT_CREATED = 'group_chat_created'
SUPERGROUP_CHAT_CREATED = 'supergroup_chat_created'
CHANNEL_CHAT_CREATED = 'channel_chat_created'
MESSAGE_AUTO_DELETE_TIMER_CHANGED = 'message_auto_delete_timer_changed'
MIGRATE_TO_CHAT_ID = 'migrate_to_chat_id'
MIGRATE_FROM_CHAT_ID = 'migrate_from_chat_id'
PINNED_MESSAGE = 'pinned_message'
INVOICE = 'invoice'
SUCCESSFUL_PAYMENT = 'successful_payment'
REFUNDED_PAYMENT = 'refunded_payment'
USERS_SHARED = 'users_shared'
CHAT_SHARED = 'chat_shared'
CONNECTED_WEBSITE = 'connected_website'
```

540

```
WRITE_ACCESS_ALLOWED = 'write_access_allowed'
    PASSPORT_DATA = 'passport_data'
    PROXIMITY_ALERT_TRIGGERED = 'proximity_alert_triggered'
    BOOST_ADDED = 'boost_added'
    CHAT_BACKGROUND_SET = 'chat_background_set'
    FORUM_TOPIC_CREATED = 'forum_topic_created'
    FORUM_TOPIC_EDITED = 'forum_topic_edited'
    FORUM_TOPIC_CLOSED = 'forum_topic_closed'
    FORUM_TOPIC_REOPENED = 'forum_topic_reopened'
    GENERAL_FORUM_TOPIC_HIDDEN = 'general_forum_topic_hidden'
    GENERAL_FORUM_TOPIC_UNHIDDEN = 'general_forum_topic_unhidden'
    GIVEAWAY_CREATED = 'giveaway_created'
    GIVEAWAY = 'giveaway'
    GIVEAWAY_WINNERS = 'giveaway_winners'
    GIVEAWAY_COMPLETED = 'giveaway_completed'
    VIDEO_CHAT_SCHEDULED = 'video_chat_scheduled'
    VIDEO_CHAT_STARTED = 'video_chat_started'
    VIDEO_CHAT_ENDED = 'video_chat_ended'
    VIDEO_CHAT_PARTICIPANTS_INVITED = 'video_chat_participants_invited'
    WEB_APP_DATA = 'web_app_data'
    USER_SHARED = 'user_shared'
Currency
class aiogram.enums.currency.currency(value, names=None, *, module=None, qualname=None,
                                       type=None, start=1, boundary=None)
    Currencies supported by Telegram Bot API
    Source: https://core.telegram.org/bots/payments#supported-currencies
    AED = 'AED'
    AFN = 'AFN'
    ALL = 'ALL'
    AMD = 'AMD'
    ARS = 'ARS'
```

- AUD = 'AUD'
- AZN = 'AZN'
- BAM = 'BAM'
- BDT = 'BDT'
- BGN = 'BGN'
- BND = 'BND'
- BOB = 'BOB'
- BRL = 'BRL'
- BYN = 'BYN'
- CAD = 'CAD'
- CHF = 'CHF'
- CLP = 'CLP'
- CNY = 'CNY'
- COP = 'COP'
- CRC = 'CRC'
- CZK = 'CZK'
- DKK = 'DKK'
- DOP = 'DOP'
- DZD = 'DZD'
- EGP = 'EGP'
- ETB = 'ETB'
- EUR = 'EUR'
- GBP = 'GBP'
- GEL = 'GEL'
- GTQ = 'GTQ'
- HKD = 'HKD'
- HNL = 'HNL'
- HRK = 'HRK'
- HUF = 'HUF'
- IDR = 'IDR'
- ILS = 'ILS'

INR = 'INR'

ISK = 'ISK'

JMD = 'JMD'

JPY = 'JPY'

KES = 'KES'

KGS = 'KGS'

KRW = 'KRW'

KZT = 'KZT'

LBP = 'LBP'

LKR = 'LKR'

MAD = 'MAD'

MDL = 'MDL'

MNT = 'MNT'

MUR = 'MUR'

MVR = 'MVR'

MXN = 'MXN'

MYR = 'MYR'

MZN = 'MZN'

NGN = 'NGN'

NIO = 'NIO'

NOK = 'NOK'

NPR = 'NPR'

NZD = 'NZD'

PAB = 'PAB'

PEN = 'PEN'

PHP = 'PHP'

PKR = 'PKR'

PLN = 'PLN'

PYG = 'PYG'

QAR = 'QAR'

RON = 'RON'

```
RSD = 'RSD'
RUB = 'RUB'
SAR = 'SAR'
SEK = 'SEK'
SGD = 'SGD'
THB = 'THB'
TJS = 'TJS'
TRY = 'TRY'
TTD = 'TTD'
TWD = 'TWD'
TZS = 'TZS'
UAH = 'UAH'
UGX = 'UGX'
USD = 'USD'
UYU = 'UYU'
UZS = 'UZS'
VND = 'VND'
YER = 'YER'
ZAR = 'ZAR'
```

DiceEmoji

Emoji on which the dice throw animation is based

Source: https://core.telegram.org/bots/api#dice
DICE = ''

DART = ''

BASKETBALL = ''

FOOTBALL = ''

SLOT_MACHINE = ''

BOWLING = ''

EncryptedPassportElement

PHOTO = 'photo'

```
class aiogram.enums.encrypted_passport_element.EncryptedPassportElement(value, names=None,
                                                                              *, module=None,
                                                                              qualname=None,
                                                                              type=None, start=1,
                                                                              boundary=None)
     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'
```

```
STICKER = 'sticker'

VIDEO = 'video'

VOICE = 'voice'

ARTICLE = 'article'

CONTACT = 'contact'

GAME = 'game'

LOCATION = 'location'

VENUE = 'venue'

InputMediaType

class aiogram.enums.input_media_type.InputMediaType(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None)

This object represents input media type
```

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

```
ANIMATION = 'animation'

AUDIO = 'audio'

DOCUMENT = 'document'

PHOTO = 'photo'

VIDEO = 'video'
```

InputPaidMediaType

This object represents the type of a media in a paid message.

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

```
PHOTO = 'photo'
VIDEO = 'video'
```

KeyboardButtonPollTypeType

class aiogram.enums.keyboard_button_poll_type_type.KeyboardButtonPollTypeType(value,

names=None,
*,
module=None,
qualname=None,
type=None,
start=1,
boundary=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

```
\begin{tabular}{ll} {\bf class} & {\it aiogram.enums.menu\_button\_type.MenuButtonType}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}
```

This object represents an type of Menu button

 $\textbf{Source:}\ https://core.telegram.org/bots/api\#menubuttonde fault$

```
DEFAULT = 'default'
COMMANDS = 'commands'
WEB_APP = 'web_app'
```

MessageEntityType

```
This object represents type of message entity
Source: https://core.telegram.org/bots/api#messageentity
MENTION = 'mention'
HASHTAG = 'hashtag'
CASHTAG = 'cashtag'
BOT_COMMAND = 'bot_command'
URL = 'url'
EMAIL = 'email'
PHONE_NUMBER = 'phone_number'
BOLD = 'bold'
ITALIC = 'italic'
UNDERLINE = 'underline'
STRIKETHROUGH = 'strikethrough'
SPOILER = 'spoiler'
BLOCKQUOTE = 'blockquote'
EXPANDABLE_BLOCKQUOTE = 'expandable_blockquote'
CODE = 'code'
```

MessageOriginType

PRE = 'pre'

TEXT_LINK = 'text_link'

TEXT_MENTION = 'text_mention'

CUSTOM_EMOJI = 'custom_emoji'

```
\begin{tabular}{ll} \textbf{class} & aiogram.enums.message\_origin\_type. \textbf{MessageOriginType}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}
```

This object represents origin of a message.

```
Source: https://core.telegram.org/bots/api#messageorigin
```

```
USER = 'user'
```

```
HIDDEN_USER = 'hidden_user'
CHAT = 'chat'
CHANNEL = 'channel'
```

PaidMediaType

This object represents the type of a media in a paid message.

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

```
PHOTO = 'photo'

PREVIEW = 'preview'

VIDEO = 'video'
```

ParseMode

Formatting options

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'
```

PollType

```
\begin{tabular}{ll} {\bf class} & {\tt aiogram.enums.poll\_type.PollType}(value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \end{tabular}
```

This object represents poll type

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

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

ReactionTypeType

This object represents reaction type.

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

```
EMOJI = 'emoji'
CUSTOM_EMOJI = 'custom_emoji'
PAID = 'paid'
```

RevenueWithdrawalStateType

class aiogram.enums.revenue_withdrawal_state_type.RevenueWithdrawalStateType(value,

names=None,
*,
module=None,
qualname=None,
type=None,
start=1, boundary=None)

This object represents a revenue withdrawal state type

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

```
FAILED = 'failed'
PENDING = 'pending'
SUCCEEDED = 'succeeded'
```

StickerFormat

```
\begin{tabular}{ll} \textbf{class} & \verb"aiogram.enums.sticker_format.StickerFormat" (value, names=None, *, module=None, qualname=None, type=None, start=1, boundary=None) \\ & Format of the sticker \end{tabular}
```

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

```
STATIC = 'static'

ANIMATED = 'animated'

VIDEO = 'video'
```

StickerType

The part of the face relative to which the mask should be placed.

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

```
REGULAR = 'regular'
MASK = 'mask'
CUSTOM_EMOJI = 'custom_emoji'
```

TopicIconColor

Color of the topic icon in RGB format.

Source: https://github.com/telegramdesktop/tdesktop/blob/991fe491c5ae62705d77aa8fdd44a79caf639c45/Telegram/SourceFiles/data/data_forum_topic.cpp#L51-L56

BLUE = 7322096 YELLOW = 16766590 VIOLET = 13338331 GREEN = 9367192 ROSE = 16749490

RED = 16478047

TransactionPartnerType

```
class aiogram.enums.transaction_partner_type.TransactionPartnerType(value, names=None, *,
                                                                         module=None,
                                                                         qualname=None,
                                                                         type=None, start=1,
                                                                         boundary=None)
     This object represents a type of transaction partner.
     Source: https://core.telegram.org/bots/api#transactionpartner
     FRAGMENT = 'fragment'
     OTHER = 'other'
     USER = 'user'
     TELEGRAM_ADS = 'telegram_ads'
     TELEGRAM_API = 'telegram_api'
     AFFILIATE_PROGRAM = 'affiliate_program'
     CHAT = 'chat'
UpdateType
class aiogram.enums.update_type.UpdateType(value, names=None, *, module=None, qualname=None,
                                              type=None, start=1, boundary=None)
     This object represents the complete list of allowed update types
     Source: https://core.telegram.org/bots/api#update
     MESSAGE = 'message'
     EDITED_MESSAGE = 'edited_message'
     CHANNEL_POST = 'channel_post'
     EDITED_CHANNEL_POST = 'edited_channel_post'
     BUSINESS_CONNECTION = 'business_connection'
     BUSINESS_MESSAGE = 'business_message'
     EDITED_BUSINESS_MESSAGE = 'edited_business_message'
     DELETED_BUSINESS_MESSAGES = 'deleted_business_messages'
     MESSAGE_REACTION = 'message_reaction'
     MESSAGE_REACTION_COUNT = 'message_reaction_count'
     INLINE_QUERY = 'inline_query'
     CHOSEN_INLINE_RESULT = 'chosen_inline_result'
     CALLBACK_QUERY = 'callback_query'
```

```
SHIPPING_QUERY = 'shipping_query'

PRE_CHECKOUT_QUERY = 'pre_checkout_query'

PURCHASED_PAID_MEDIA = 'purchased_paid_media'

POLL = 'poll'

POLL_ANSWER = 'poll_answer'

MY_CHAT_MEMBER = 'my_chat_member'

CHAT_MEMBER = 'chat_member'

CHAT_JOIN_REQUEST = 'chat_join_request'

CHAT_BOOST = 'chat_boost'

REMOVED_CHAT_BOOST = 'removed_chat_boost'
```

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: $str \mid Path$, destination: BinaryIO | Path | $str \mid None = None$, timeout: int = 30, $chunk_size$: int = 65536, seek: bool = True) \rightarrow BinaryIO | None

Download file by file_path to destination.

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

Parameters

- **file_path** File path on Telegram server (You can get it from aiogram.types.File)
- **destination** Filename, file path or instance of io.IOBase. For e.g. io.BytesIO, defaults to None

- **timeout** Total timeout in seconds, defaults to 30
- chunk_size File chunks size, defaults to 64 kb
- **seek** Go to start of file when downloading is finished. Used only for destination with typing.BinaryIO type, defaults to True

There are two options where you can download the file: to disk or to binary I/O object.

Download file to disk

To download file to disk, you must specify the file name or path where to download the file. In this case, the function will return nothing.

```
await bot.download_file(file_path, "text.txt")
```

Download file to binary I/O object

To download file to binary I/O object, you must specify an object with the typing.BinaryIO type or use the default (None) value.

In the first case, the function will return your object:

```
my_object = MyBinaryIO()
result: MyBinaryIO = await bot.download_file(file_path, my_object)
# print(result is my_object) # True
```

If you leave the default value, an io. BytesIO object will be created and returned.

```
result: io.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

A Warning

Be respectful to Telegram

Instances of *InputFile* are reusable. That means you can create an instance of InputFile and send it multiple times. However, Telegram does not recommend doing this. Instead, once you upload a file, save its *file_id* and reuse that later.

Upload from file system

By first step you will need to import InputFile wrapper:

```
from aiogram.types import FSInputFile
```

Then you can use it:

```
cat = FSInputFile("cat.png")
agenda = FSInputFile("my-document.pdf", filename="agenda-2019-11-19.pdf")
```

class aiogram.types.input_file.FSInputFile($path: str \mid Path, filename: str \mid None = None, chunk_size: int = 65536$)

```
__init__(path: str | Path, filename: str | None = None, chunk_size: int = 65536)

Represents object for uploading files from filesystem
```

Parameters

- path Path to file
- filename Filename to be propagated to telegram. By default, will be parsed from path
- **chunk_size** Uploading chunk size

Upload from buffer

Files can be also passed from buffer (For example you generate image using Pillow and you want to send it to Telegram): Import wrapper:

```
from aiogram.types import BufferedInputFile
```

And then you can use it:

```
text_file = BufferedInputFile(b"Hello, world!", filename="file.txt")
```

class aiogram.types.input_file.**BufferedInputFile**(*file: bytes, filename: str, chunk_size: int = 65536*)

```
__init__(file: bytes, filename: str, chunk_size: int = 65536)
```

Represents object for uploading files from filesystem

Parameters

- file Bytes
- **filename** Filename to be propagated to telegram.
- **chunk_size** Uploading chunk size

Upload from url

If you need to upload a file from another server, but the direct link is bound to your server's IP, or you want to bypass native upload limits by URL, you can use <code>aiogram.types.input_file.URLInputFile</code>.

Import wrapper:

```
from aiogram.types import URLInputFile
```

And then you can use it:

```
image = URLInputFile(
    "https://www.python.org/static/community_logos/python-powered-h-140x182.png",
    filename="python-logo.png"
)
```

2.3.8 Global defaults

aiogram provides mechanism to set some global defaults for all requests to Telegram Bot API in your application using aiogram.client.default.DefaultBotProperties class.

There are some properties that can be set:

parse_mode: str | None

Default parse mode for messages.

disable_notification: bool | None

Sends the message silently. Users will receive a notification with no sound.

protect_content: bool | None Protects content from copying.

allow_sending_without_reply: bool | None

Allows to send messages without reply.

link_preview: LinkPreviewOptions | None

Link preview settings.

link_preview_is_disabled: bool | None

Disables link preview.

link_preview_prefer_small_media: bool | None

Prefer small media in link preview.

link_preview_prefer_large_media: bool | None

Prefer large media in link preview.

link_preview_show_above_text: bool | None

Show link preview above text.

show_caption_above_media: bool | None

Show caption above media.



If you need to override default properties for some requests, you should use aiogram.client.default.DefaultBotProperties only for properties that you want to set as defaults and pass explicit values for other properties.

Danger

If you upgrading from aiogram 3.0-3.6 to 3.7, you should update your code to use aiogram.client.default.DefaultBotProperties.

Example

Here is an example of setting default parse mode for all requests to Telegram Bot API:

```
bot = Bot(
    token=...,
    default=DefaultBotProperties(
        parse_mode=ParseMode.HTML,
    )
)
```

In this case all messages sent by this bot will be parsed as HTML, so you don't need to specify *parse_mode* in every message you send.

Instead of

```
await bot.send_message(chat_id, text, parse_mode=ParseMode.HTML)
```

you can use

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

and the message will be sent with HTML parse mode.

In some cases you may want to override default properties for some requests. You can do it by passing explicit values to the method:

```
await bot.send_message(chat_id, text, parse_mode=ParseMode.MARKDOWN_V2)
```

In this case the message will be sent with Markdown parse mode instead of default HTML.

Another example of overriding default properties:

```
await bot.send_message(chat_id, text, parse_mode=None)
```

In this case the message will be send without parse mode, even if default parse mode is set it may be useful if you want to send message with plain text or aiogram.types.message_entity.MessageEntity.

```
await bot.send_message(
    chat_id=chat_id,
    text=text,
    entities=[MessageEntity(type='bold', offset=0, length=4)],
    parse_mode=None
)
```

2.4 Handling events

aiogram includes Dispatcher mechanism. Dispatcher is needed for handling incoming updates from Telegram.

With dispatcher you can do:

- Handle incoming updates;
- Filter incoming events before it will be processed by specific handler;
- · Modify event and related data in middlewares;
- Separate bot functionality between different handlers, modules and packages

Dispatcher is also separated into two entities - Router and Dispatcher. Dispatcher is subclass of router and should be always is root router.

Telegram supports two ways of receiving updates:

- Webhook you should configure your web server to receive updates from Telegram;
- Long polling you should request updates from Telegram.

So, you can use both of them with aiogram.

2.4.1 Router

Usage:

```
from aiogram import Router
from aiogram.types import Message

my_router = Router(name=__name__)

@my_router.message()
async def message_handler(message: Message) -> Any:
    await message.answer('Hello from my router!')
```

```
class aiogram.dispatcher.router.Router(*, name: str | None = None)
```

Bases: object

Router can route update, and it nested update types like messages, callback query, polls and all other event types.

Event handlers can be registered in observer by two ways:

```
    By observer method - router.<event_type>.register(handler, <filters, ...>)
    By decorator - @router.<event_type>(<filters, ...>)
    __init__(*, name: str | None = None) → None
    Parameters
        name - Optional router name, can be useful for debugging
```

 $include_router(router: Router) \rightarrow Router$

Attach another router.

Parameters router

Returns

```
include_routers(*routers: Router) → None
     Attach multiple routers.
         Parameters
             routers
         Returns
resolve\_used\_update\_types(skip\_events: Set[str] | None = None) \rightarrow List[str]
     Resolve registered event names
     Is useful for getting updates only for registered event types.
         Parameters
              skip_events – skip specified event names
         Returns
              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

```
@router.message_reaction_count()
async def message_reaction_count_handler(message_reaction_count: types.
→MessageReactionCountUpdated) -> Any: pass
```

Chat boost

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

Remove chat boost

```
@router.removed_chat_boost()
async def removed_chat_boost_handler(removed_chat_boost: types.ChatBoostRemoved) -> Any:_
→pass
```

Errors

```
@router.errors()
async def error_handler(exception: types.ErrorEvent) -> Any: pass
```

Is useful for handling errors from other handlers, error event described *here*

Nested routers



Warning

Routers by the way can be nested to an another routers with some limitations:

1. Router CAN NOT include itself 1. Routers CAN NOT be used for circular including (router 1 include router 2, router 2 include router 3, router 3 include router 1)

Example:

```
Listing 1: module_1.py
```

name

```
module_1
router2 = Router()
@router2.message() ...
```

Listing 2: module_2.py

name

 $module_2$

from module_2 import router2

router1 = Router() router1.include_router(router2)

Update

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

▲ Warning

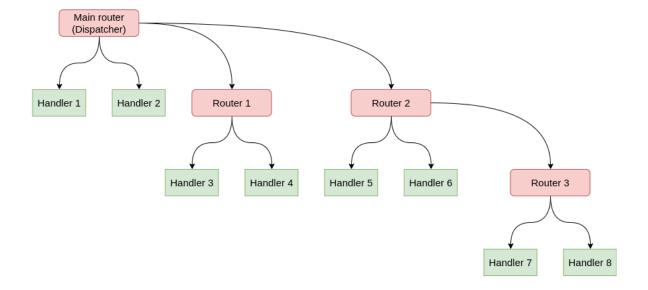
The only root Router (Dispatcher) can handle this type of event.

1 Note

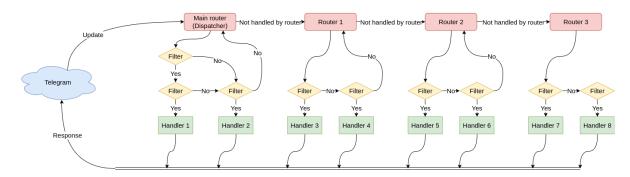
Dispatcher already has default handler for this event type, so you can use it for handling all updates that are not handled by any other handlers.

How it works?

For example, dispatcher has 2 routers, the last router also has one nested router:



In this case update propagation flow will have form:



2.4.2 Dispatcher

Dispatcher is root *Router* and in code Dispatcher can be used directly for routing updates or attach another routers into dispatcher.

Here is only listed base information about Dispatcher. All about writing handlers, filters and etc. you can find in next pages:

- Router
- Filtering events

Root router

```
__init__(*, storage: BaseStorage | None = None, fsm_strategy: FSMStrategy = FSMStrategy.USER_IN_CHAT, events_isolation: BaseEventIsolation | None = None, disable_fsm: bool = False, name: str | None = None, **kwargs: Any) → None
```

Root router

Parameters

- storage Storage for FSM
- **fsm_strategy** FSM strategy
- events_isolation Events isolation
- **disable_fsm** Disable FSM, note that if you disable FSM then you should not use storage and events isolation
- **kwargs** Other arguments, will be passed as keyword arguments to handlers

 $\textbf{async feed_raw_update}(\textit{bot: Bot, update: Dict[str, Any]}, **kwargs: Any) \rightarrow Any$

Main entry point for incoming updates with automatic Dict->Update serializer

Parameters

- bot
- update

kwargs

```
async feed_update(bot: Bot, update: Update, **kwargs: Any) \rightarrow Any
```

Main entry point for incoming updates Response of this method can be used as Webhook response

Parameters

- bot
- update

```
 \begin{aligned} \textbf{run\_polling}(*bots: Bot, polling\_timeout: int = 10, handle\_as\_tasks: bool = True, backoff\_config: \\ BackoffConfig = BackoffConfig(min\_delay=1.0, max\_delay=5.0, factor=1.3, jitter=0.1), \\ allowed\_updates: List[str] \mid \_SentinelObject \mid None = sentinel.UNSET, handle\_signals: bool \\ = True, close\_bot\_session: bool = True, **kwargs: Any) \rightarrow \text{None} \end{aligned}
```

Run many bots with polling

Parameters

- **bots** Bot instances (one or more)
- **polling_timeout** Long-polling wait time
- handle_as_tasks Run task for each event and no wait result
- backoff_config backoff-retry config
- allowed_updates List of the update types you want your bot to receive
- handle_signals handle signals (SIGINT/SIGTERM)
- close_bot_session close bot sessions on shutdown
- kwargs contextual data

Returns

```
async start_polling(*bots: Bot, polling_timeout: int = 10, handle_as_tasks: bool = True, backoff_config: BackoffConfig = BackoffConfig(min_delay=1.0, max_delay=5.0, factor=1.3, jitter=0.1), allowed_updates: List[str] | _SentinelObject | None = sentinel.UNSET, handle_signals: bool = True, close_bot_session: bool = True, **kwargs: Any) \rightarrow None
```

Polling runner

Parameters

- **bots** Bot instances (one or more)
- polling_timeout Long-polling wait time
- handle_as_tasks Run task for each event and no wait result
- backoff_config backoff-retry config
- **allowed_updates** List of the update types you want your bot to receive By default, all used update types are enabled (resolved from handlers)
- handle_signals handle signals (SIGINT/SIGTERM)
- close_bot_session close bot sessions on shutdown
- **kwargs** contextual data

Returns

```
async stop_polling() \rightarrow None
```

Execute this method if you want to stop polling programmatically

Returns

Simple usage

Example:

```
dp = Dispatcher()

@dp.message()
async def message_handler(message: types.Message) -> None:
    await SendMessage(chat_id=message.from_user.id, text=message.text)
```

Including routers

Example:

```
dp = Dispatcher()
router1 = Router()
dp.include_router(router1)
```

Handling updates

All updates can be propagated to the dispatcher by feed_update() method:

```
from aiogram import Bot, Dispatcher
async def update_handler(update: Update, bot: Bot, dispatcher: Dispatcher):
    result = await dp.feed_update(bot, update)
```

Also you can feed raw update (dictionary) object to the dispatcher by feed_raw_update() method:

```
from aiogram import Bot, Dispatcher

async def update_handler(raw_update: dict[str, Any], bot: Bot, dispatcher: Dispatcher):
    result = await dp.feed_raw_update(bot, raw_update)
```

2.4.3 Dependency injection

Dependency injection is a programming technique that makes a class independent of its dependencies. It achieves that by decoupling the usage of an object from its creation. This helps you to follow SOLID's dependency inversion and single responsibility principles.

How it works in aiogram

For each update aiogram.dispatcher.dispatcher.Dispatcher passes handling context data. Filters and middleware can also make changes to the context.

To access contextual data you should specify corresponding keyword parameter in handler or filter. For example, to get aiogram.fsm.context.FSMContext we do it like that:

```
@router.message(ProfileCompletion.add_photo, F.photo)
async def add_photo(
   message: types.Message, bot: Bot, state: FSMContext
) -> Any:
   ... # do something with photo
```

Injecting own dependencies

Aiogram provides several ways to complement / modify contextual data.

The first and easiest way is to simply specify the named arguments in aiogram.dispatcher.dispatcher.Dispatcher initialization, polling start methods or aiogram.webhook.aiohttp_server.SimpleRequestHandler initialization if you use webhooks.

```
async def main() -> None:
    dp = Dispatcher(..., foo=42)
    return await dp.start_polling(
        bot, bar="Bazz"
)
```

Analogy for webhook:

```
async def main() -> None:
    dp = Dispatcher(..., foo=42)
    handler = SimpleRequestHandler(dispatcher=dp, bot=bot, bar="Bazz")
    ... # starting webhook
```

<code>aiogram.dispatcher.Dispatcher</code>'s workflow data also can be supplemented by setting values as in a dictionary:

```
dp = Dispatcher(...)
dp["eggs"] = Spam()
```

The middlewares updates the context quite often. You can read more about them on this page:

Middlewares

The last way is to return a dictionary from the filter:

```
from typing import Any, Dict, Optional, Union

from aiogram import Router
from aiogram.filters import Filter
from aiogram.types import Message, User

router = Router(name=__name__)
```

(continues on next page)

(continued from previous page)

```
class HelloFilter(Filter):
   def __init__(self, name: Optional[str] = None) -> None:
        self.name = name
    async def __call__(
        self,
       message: Message,
       event_from_user: User,
        # Filters also can accept keyword parameters like in handlers
   ) -> Union[bool, Dict[str, Any]]:
       if message.text.casefold() == "hello":
            # Returning a dictionary that will update the context data
            return {"name": event_from_user.mention_html(name=self.name)}
       return False
@router.message(HelloFilter())
async def my_handler(
   message: Message, name: str # Now we can accept "name" as named parameter
   return message.answer("Hello, {name}!".format(name=name))
```

...or using *MagicFilter* with .as_(...) method.

Using type hints



Note

Type-hinting middleware data is optional and is not required for the correct operation of the dispatcher. However, it is recommended to use it to improve the readability of the code.

You can use type hints to specify the type of the context data in the middlewares, filters and handlers.

The default middleware data typed dict can be found in aiogram.dispatcher.middlewares.data. MiddlewareData.

In case when you have extended the context data, you can use the aiogram.dispatcher.middlewares.data. MiddlewareData as a base class and specify the type hints for the new fields.

Warning

If you using type checking tools like mypy, you can experience warnings about that this type hint against Liskov substitution principle in due stricter type is not a subclass of dict[str, Any]. This is a known issue and it is not a bug. You can ignore this warning or use # type: ignore comment.

Example of using type hints:

```
from aiogram.dispatcher.middlewares.data import MiddlewareData
                                                                                    (continues on next page)
```

(continued from previous page)

```
class MyMiddlewareData(MiddlewareData, total=False):
    my_custom_value: int

class MyMessageMiddleware(BaseMiddleware):
    async def __call__(
        self,
        handler: Callable[[Message, MyMiddlewareData], Awaitable[Any]],
        event: Message,
        data: MyMiddlewareData,
    ) -> Any:
        bot = data["bot"] # <-- IDE will show you that data has `bot` key and its type__
        is `Bot`

        data["my_custom_value"] = bot.id * 42 # <-- IDE will show you that you can set__
        'my_custom_value` key with int value and warn you if you try to set it with other type
        return await handler(event, data)</pre>
```

Available context data type helpers

```
class aiogram.dispatcher.middlewares.data.MiddlewareData
    Data passed to the handler by the middlewares.
    You can add your own data by extending this class.
    dispatcher: Dispatcher
    bot: Bot
    bots: NotRequired[list[Bot]]
    event_update: Update
    event_router: Router
    handler: NotRequired[HandlerObject]
    event_context: EventContext
    event_from_user: NotRequired[User]
    event_chat: NotRequired[Chat]
    event_thread_id: NotRequired[int]
    event_business_connection_id: NotRequired[str]
    fsm_storage: BaseStorage
    state: NotRequired[FSMContext]
    raw_state: NotRequired[str | None]
```

class aiogram.dispatcher.middlewares.data.I18nData

I18n related data.

Is not included by default, you need to add it to your own Data class if you need it.

i18n: I18n I18n object.

i18n_middleware: I18nMiddleware

I18n middleware.

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")

A Warning

Command cannot include spaces or any whitespace

```
 \textbf{class} \  \, \textbf{aiogram.filters.command}. \textbf{Command}(*values: str \mid Pattern \mid BotCommand, commands: Sequence[str \mid Pattern \mid BotCommand] \mid str \mid Pattern \mid BotCommand \mid None = None, prefix: str = '/', ignore_case: bool = False, ignore_mention: bool = False, magic: MagicFilter | 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 \mid None = None, args: str \mid None = None, regexp\_match: Match[str] \mid None = None, magic\_result: Any \mid 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 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



This page still in progress. Has many incorrectly worded sentences.

Is external package maintained by aiogram core team.

By default installs with aiogram and also is available on PyPi - magic-filter. That's mean you can install it and use with any other libraries and in own projects without depending aiogram installed.

Usage

The magic_filter package implements class shortly named magic_filter.F that's mean F can be imported from aiogram or magic_filter. F is alias for MagicFilter.



Note that aiogram has an small extension over magic-filter and if you want to use this extension you should import magic from aiogram instead of magic filter package

The MagicFilter object is callable, supports some actions and memorize the attributes chain and the action which should be checked on demand.

So that's mean you can chain attribute getters, describe simple data validations and then call the resulted object passing single object as argument, for example make attributes chain F. foo.bar.baz then add action 'F. foo.bar.baz == 'spam' and then call the resulted object - (F.foo.bar.baz == 'spam').resolve(obj)

Possible actions

Magic filter object supports some of basic logical operations over object attributes

Exists or not None

Default actions.

```
F.photo # lambda message: message.photo
```

Equals

```
F.text == 'hello' # lambda message: message.text == 'hello'
F.from_user.id == 42 # lambda message: message.from_user.id == 42
F.text != 'spam' # lambda message: message.text != 'spam'
```

Is one of

Can be used as method named in_ or as matmul operator @ with any iterable

```
F.from_user.id.in_({42, 1000, 123123}) # lambda query: query.from_user.id in {42, 1000, _
→ 123123}
F.data.in_({'foo', 'bar', 'baz'}) # lambda query: query.data in {'foo', 'bar', 'baz'}
```

Contains

```
F.text.contains('foo') # lambda message: 'foo' in message.text
```

String startswith/endswith

Can be applied only for text attributes

```
F.text.startswith('foo') # lambda message: message.text.startswith('foo')
F.text.endswith('bar') # lambda message: message.text.startswith('bar')
```

Regexp

```
F.text.regexp(r'Hello, .+') # lambda message: re.match(r'Hello, .+', message.text)
```

Custom function

Accepts any callable. Callback will be called when filter checks result

```
F.chat.func(lambda chat: chat.id == -42) # lambda message: (lambda chat: chat.id == -42) (message.chat)
```

Inverting result

Any of available operation can be inverted by bitwise inversion - ~

```
~F.text # lambda message: not message.text
~F.text.startswith('spam') # lambda message: not message.text.startswith('spam')
```

Combining

All operations can be combined via bitwise and/or operators - &/|

Attribute modifiers - string manipulations

Make text upper- or lower-case

Can be used only with string attributes.

```
F.text.lower() == 'test' # lambda message: message.text.lower() == 'test'
F.text.upper().in_({'F00', 'BAR'}) # lambda message: message.text.upper() in {'F00', 'BAR'}
F.text.len() == 5 # lambda message: len(message.text) == 5
```

Get filter result as handler argument

This part is not available in magic-filter directly but can be used with aiogram

```
from aiogram import F
...
@router.message(F.text.regexp(r"^(\d+)$").as_("digits"))
async def any_digits_handler(message: Message, digits: Match[str]):
    await message.answer(html.quote(str(digits)))
```

Usage in aiogram

```
@router.message(F.text == 'hello')
@router.inline_query(F.data == 'button:1')
@router.message(F.text.startswith('foo'))
@router.message(F.content_type.in_({\text', 'sticker'}))
@router.message(F.text.regexp(r'\d+'))
...
# Many others cases when you will need to check any of available event attribute
```

MagicData

Usage

1. MagicData(F.event.from_user.id == F.config.admin_id) (Note that config should be passed from middleware)

Explanation

```
{\bf class}\ {\bf aiogram.filters.magic\_data.MagicData} ({\it 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
```

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:

```
builder = InlineKeyboardBuilder()
builder.button(
    text="demo",
    callback_data=MyCallback(foo="demo", bar="42") # Value can be not packed to string_
    inplace, because builder knows what to do with callback instance
)
```

Another abstract example:

```
class Action(str, Enum):
   ban = "ban"
   kick = "kick"
   warn = "warn"
class AdminAction(CallbackData, prefix="adm"):
   action: Action
   chat_id: int
   user id: int
# Inside handler
builder = InlineKeyboardBuilder()
for action in Action:
   builder.button(
        text=action.value.title(),
        callback_data=AdminAction(action=action, chat_id=chat_id, user_id=user_id),
await bot.send_message(
   chat_id=admins_chat,
   text=f"What do you want to do with {html.quote(name)}",
   reply_markup=builder.as_markup(),
)
@router.callback_query(AdminAction.filter(F.action == Action.ban))
async def ban_user(query: CallbackQuery, callback_data: AdminAction, bot: Bot):
   await bot.ban_chat_member(
        chat_id=callback_data.chat_id,
       user_id=callback_data.user_id,
   )
```

Known limitations

Allowed types and their subclasses:

- str
- int
- bool
- float
- Decimal (from decimal import Decimal)

- Fraction (from fractions import Fraction)
- UUID (from uuid import UUID)
- Enum (from enum import Enum, only for string enums)
- IntEnum (from enum import IntEnum, only for int enums)

1 Note

Note that the integer Enum's should be always is subclasses of IntEnum in due to parsing issues.

Exceptions

This filters can be helpful for handling errors from the text messages.

```
\begin{cal}{class aiogram.filters.exception.} \textbf{ExceptionTypeFilter} (*exceptions: Type[Exception]) \\ Allows to match exception by type \end{cal}
```

exceptions

```
class aiogram.filters.exception.ExceptionMessageFilter(pattern: str | Pattern[str])
Allow to match exception by message
```

pattern

Allowed handlers

Allowed update types for this filters:

error

Writing own filters

Filters can be:

- Asynchronous function (async def my_filter(*args, **kwargs): pass)
- Synchronous function (def my_filter(*args, **kwargs): pass)
- Anonymous function (lambda event: True)
- · Any awaitable object
- Subclass of aiogram.filters.base.Filter
- Instances of MagicFilter

and should return bool or dict. If the dictionary is passed as result of filter - resulted data will be propagated to the next filters and handler as keywords arguments.

Base class for own filters

```
class aiogram.filters.base.Filter
```

If you want to register own filters like builtin filters you will need to write subclass of this class with overriding the __call__ method and adding filter attributes.

```
abstract async __call__(*args: Any, **kwargs: Any) → bool | Dict[str, Any]
```

This method should be overridden.

Accepts incoming event and should return boolean or dict.

Returns

bool or Dict[str, Any]

```
update_handler_flags(flags: Dict[str, Any]) \rightarrow None
```

Also if you want to extend handler flags with using this filter you should implement this method

Parameters

flags – existing flags, can be updated directly

Own filter example

For example if you need to make simple text filter:

```
from aiogram import Router
from aiogram.filters import Filter
from aiogram.types import Message

router = Router()

class MyFilter(Filter):
    def __init__(self, my_text: str) -> None:
        self.my_text = my_text

    async def __call__(self, message: Message) -> bool:
        return message.text == self.my_text

Grouter.message(MyFilter("hello"))
async def my_handler(message: Message): ...
```

Combining Filters

In general, all filters can be combined in two ways

Recommended way

If you specify multiple filters in a row, it will be checked with an "and" condition:

```
@<router>.message(F.text.startswith("show"), F.text.endswith("example"))
```

Also, if you want to use two alternative ways to run the same handler ("or" condition) you can register the handler twice or more times as you like

```
@<router>.message(F.text == "hi")
@<router>.message(CommandStart())
```

Also sometimes you will need to invert the filter result, for example you have an *IsAdmin* filter and you want to check if the user is not an admin

```
@<router>.message(~IsAdmin())
```

Another possible way

An alternative way is to combine using special functions $(and_f(), or_f(), invert_f())$ from aiogram. filters module):

```
and_f(F.text.startswith("show"), F.text.endswith("example"))
or_f(F.text(text="hi"), CommandStart())
invert_f(IsAdmin())
and_f(<A>, or_f(<B>, <C>))
```

2.4.5 Long-polling

Long-polling is a technology that allows a Telegram server to send updates in case when you don't have dedicated IP address or port to receive webhooks for example on a developer machine.

To use long-polling mode you should use aiogram.dispatcher.dispatcher.Dispatcher.start_polling() or aiogram.dispatcher.Dispatcher.run_polling() methods.

1 Note

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

1 Note

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

1 Note

If you will use multibot mode, you should use webhook mode for all bots.

Example

This example will show you how to create simple echo bot based on long-polling.

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

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

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

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

2.4.6 Finite State Machine

A finite-state machine (FSM) or finite-state automaton (FSA, plural: automata), finite automaton, or simply a state machine, is a mathematical model of computation.

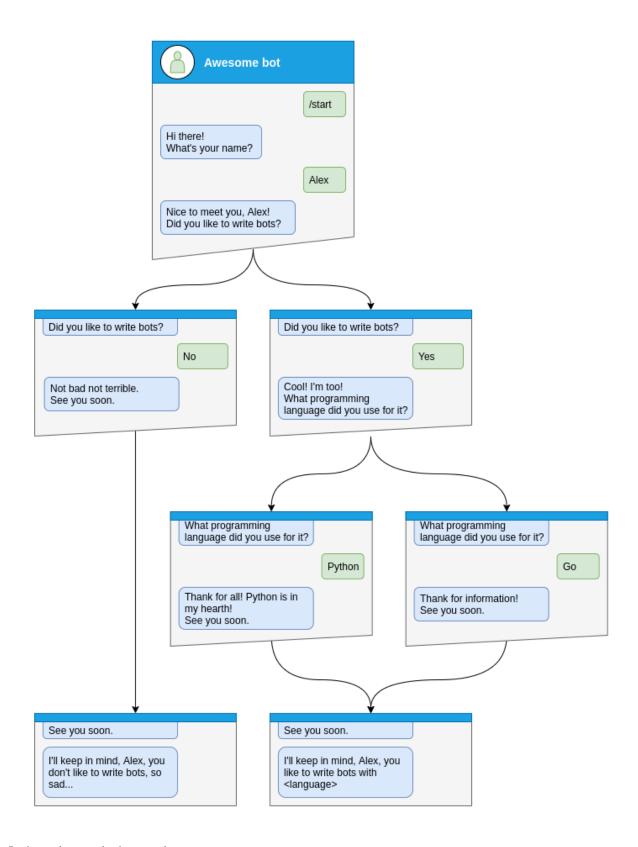
It is an abstract machine that can be in exactly one of a finite number of states at any given time. The FSM can change from one state to another in response to some inputs; the change from one state to another is called a transition.

An FSM is defined by a list of its states, its initial state, and the inputs that trigger each transition.

Source: WikiPedia

Usage example

Not all functionality of the bot can be implemented as single handler, for example you will need to collect some data from user in separated steps you will need to use FSM.



Let's see how to do that step-by-step

586 Chapter 2. Contents

Step by step

Before handle any states you will need to specify what kind of states you want to handle

```
class Form(StatesGroup):
   name = State()
   like_bots = State()
   language = State()
```

And then write handler for each state separately from the start of dialog

Here is dialog can be started only via command /start, so lets handle it and make transition user to state Form.name

```
@form_router.message(CommandStart())
async def command_start(message: Message, state: FSMContext) -> None:
    await state.set_state(Form.name)
    await message.answer(
        "Hi there! What's your name?",
        reply_markup=ReplyKeyboardRemove(),
    )
```

After that you will need to save some data to the storage and make transition to next step.

```
@form_router.message(Form.name)
async def process_name(message: Message, state: FSMContext) -> None:
    await state.update_data(name=message.text)
    await state.set_state(Form.like_bots)
   await message.answer(
        f"Nice to meet you, {html.quote(message.text)}!\nDid you like to write bots?",
        reply_markup=ReplyKeyboardMarkup(
            keyboard=[
                Ε
                    KeyboardButton(text="Yes"),
                    KeyboardButton(text="No"),
                ]
            ],
            resize_keyboard=True,
        ),
   )
```

At the next steps user can make different answers, it can be yes, no or any other

Handle yes and soon we need to handle Form.language state

```
@form_router.message(Form.like_bots, F.text.casefold() == "yes")
async def process_like_write_bots(message: Message, state: FSMContext) -> None:
    await state.set_state(Form.language)

await message.reply(
    "Cool! I'm too!\nWhat programming language did you use for it?",
    reply_markup=ReplyKeyboardRemove(),
)
```

Handle no

```
@form_router.message(Form.like_bots, F.text.casefold() == "no")
async def process_dont_like_write_bots(message: Message, state: FSMContext) -> None:
    data = await state.get_data()
    await state.clear()
    await message.answer(
        "Not bad not terrible.\nSee you soon.",
        reply_markup=ReplyKeyboardRemove(),
    )
    await show_summary(message=message, data=data, positive=False)
```

And handle any other answers

```
@form_router.message(Form.like_bots)
async def process_unknown_write_bots(message: Message) -> None:
    await message.reply("I don't understand you :(")
```

All possible cases of like_bots step was covered, let's implement finally step

```
async def show_summary(message: Message, data: Dict[str, Any], positive: bool = True) ->_
None:
    name = data["name"]
    language = data.get("language", "<something unexpected>")
    text = f"I'll keep in mind that, {html.quote(name)}, "
    text += (
        f"you like to write bots with {html.quote(language)}."
        if positive
        else "you don't like to write bots, so sad..."
    )
    await message.answer(text=text, reply_markup=ReplyKeyboardRemove())
```

And now you have covered all steps from the image, but you can make possibility to cancel conversation, lets do that via command or text

```
@form_router.message(Command("cancel"))
@form_router.message(F.text.casefold() == "cancel")
async def cancel_handler(message: Message, state: FSMContext) -> None:
    """
    Allow user to cancel any action
    """
    current_state = await state.get_state()
    if current_state is None:
```

```
return

logging.info("Cancelling state %r", current_state)
await state.clear()
await message.answer(
    "Cancelled.",
    reply_markup=ReplyKeyboardRemove(),
)
```

Complete example

```
import asyncio
   import logging
   import sys
   from os import getenv
   from typing import Any, Dict
   from aiogram import Bot, Dispatcher, F, Router, html
   from aiogram.client.default import DefaultBotProperties
   from aiogram.enums import ParseMode
   from aiogram.filters import Command, CommandStart
10
   from aiogram.fsm.context import FSMContext
11
   from aiogram.fsm.state import State, StatesGroup
12
   from aiogram.types import (
13
       KeyboardButton,
14
       Message,
       ReplyKeyboardMarkup,
16
       ReplyKeyboardRemove,
   )
18
   TOKEN = getenv("BOT_TOKEN")
20
21
   form_router = Router()
22
23
24
   class Form(StatesGroup):
25
       name = State()
26
       like_bots = State()
27
       language = State()
28
29
   @form_router.message(CommandStart())
31
   async def command_start(message: Message, state: FSMContext) -> None:
32
       await state.set_state(Form.name)
33
       await message.answer(
34
           "Hi there! What's your name?",
35
           reply_markup=ReplyKeyboardRemove(),
       )
37
38
```

```
@form_router.message(Command("cancel"))
40
   @form_router.message(F.text.casefold() == "cancel")
   async def cancel_handler(message: Message, state: FSMContext) -> None:
42
       Allow user to cancel any action
44
       current_state = await state.get_state()
46
       if current_state is None:
47
           return
48
49
       logging.info("Cancelling state %r", current_state)
50
       await state.clear()
51
       await message.answer(
52
            "Cancelled.",
53
           reply_markup=ReplyKeyboardRemove(),
       )
55
57
   @form_router.message(Form.name)
   async def process_name(message: Message, state: FSMContext) -> None:
59
       await state.update_data(name=message.text)
       await state.set state(Form.like bots)
61
       await message.answer(
62
           f"Nice to meet you, {html.quote(message.text)}!\nDid you like to write bots?",
63
           reply_markup=ReplyKeyboardMarkup(
                keyboard=[
65
                    Ε
                        KeyboardButton(text="Yes"),
                        KeyboardButton(text="No"),
68
                ],
70
                resize_keyboard=True,
           ),
72
       )
74
   @form_router.message(Form.like_bots, F.text.casefold() == "no")
76
   async def process_dont_like_write_bots(message: Message, state: FSMContext) -> None:
       data = await state.get_data()
78
       await state.clear()
       await message.answer(
80
           "Not bad not terrible.\nSee you soon.",
81
           reply_markup=ReplyKeyboardRemove(),
82
83
       await show_summary(message=message, data=data, positive=False)
84
85
   @form_router.message(Form.like_bots, F.text.casefold() == "yes")
87
   async def process_like_write_bots(message: Message, state: FSMContext) -> None:
       await state.set_state(Form.language)
89
       await message reply(
91
```

```
"Cool! I'm too!\nWhat programming language did you use for it?",
92
            reply_markup=ReplyKeyboardRemove(),
        )
    @form_router.message(Form.like_bots)
    async def process_unknown_write_bots(message: Message) -> None:
        await message.reply("I don't understand you :(")
100
101
   @form_router.message(Form.language)
102
    async def process_language(message: Message, state: FSMContext) -> None:
103
        data = await state.update_data(language=message.text)
        await state.clear()
105
        if message.text.casefold() == "python":
107
            await message.reply(
                "Python, you say? That's the language that makes my circuits light up! "
109
110
111
        await show_summary(message=message, data=data)
112
113
114
   async def show_summary(message: Message, data: Dict[str, Any], positive: bool = True) ->_
115
    None:
        name = data["name"]
116
        language = data.get("language", "<something unexpected>")
117
        text = f"I'll keep in mind that, {html.quote(name)}, "
118
        text += (
119
            f"you like to write bots with {html.quote(language)}."
120
            if positive
121
            else "you don't like to write bots, so sad..."
122
123
        await message.answer(text=text, reply_markup=ReplyKeyboardRemove())
125
    async def main():
127
        # Initialize Bot instance with default bot properties which will be passed to all
        bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))
129
130
        dp = Dispatcher()
131
132
        dp.include_router(form_router)
133
134
        # Start event dispatching
135
        await dp.start_polling(bot)
137
    if __name__ == "__main__":
139
        logging.basicConfig(level=logging.INFO, stream=sys.stdout)
140
        asyncio.run(main())
141
```

Read more

Storages

Storages out of the box

MemoryStorage

class aiogram.fsm.storage.memory.MemoryStorage

Default FSM storage, stores all data in dict and loss everything on shutdown



Warning

Is not recommended using in production in due to you will lose all data when your bot restarts

```
\_init\_() \rightarrow None
```

RedisStorage

```
class aiogram.fsm.storage.redis.RedisStorage(redis: ~redis.asyncio.client.Redis, key_builder:
                                                         ~aiogram.fsm.storage.base.KeyBuilder | None = None,
                                                         state\_ttl: int \mid \sim datetime.timedelta \mid None = None,
                                                         data ttl: int | ~datetime.timedelta | None = None,
                                                         json_loads: ~typing.Callable[[...], ~typing.Any] =
                                                          <function loads>, json_dumps: ~typing.Callable[[...],
                                                         str = < function dumps>)
```

Redis storage required redis package installed (pip install redis)

```
__init__(redis: ~redis.asyncio.client.Redis, key_builder: ~aiogram.fsm.storage.base.KeyBuilder | None =
            None, state_ttl: int | ~datetime.timedelta | None = None, data_ttl: int | ~datetime.timedelta | None
            = None, json\_loads: \sim typing.Callable[[...], \sim typing.Any] = < function loads>, <math>json\_dumps:
            ~typing.Callable[[...], str] = <function dumps>) \rightarrow None
```

Parameters

- redis Instance of Redis connection
- **key_builder** builder that helps to convert contextual key to string
- state_ttl TTL for state records
- data_ttl TTL for data records

classmethod from_url(url: str, connection_kwargs: $Dict[str, Any] \mid None = None, **kwargs: Any) \rightarrow$ RedisStorage

Create an instance of *RedisStorage* with specifying the connection string

Parameters

- url for example redis://user:password@host:port/db
- connection_kwargs see redis docs
- **kwargs** arguments to be passed to *RedisStorage*

Returns

an instance of RedisStorage

MongoStorage

```
class aiogram.fsm.storage.mongo.MongoStorage(client: AsyncIOMotorClient, key_builder: KeyBuilder| None = None, db_name: str = 'aiogram\_fsm', collection\_name: str = 'states\_and\_data')
```

MongoDB storage required motor package installed (pip install motor)

```
__init__(client: AsyncIOMotorClient, key_builder: KeyBuilder | None = None, db_name: str = 'aiogram\_fsm', collection_name: str = 'states\_and\_data') \rightarrow None
```

Parameters

- **client** Instance of AsyncIOMotorClient
- **key_builder** builder that helps to convert contextual key to string
- db_name name of the MongoDB database for FSM
- collection_name name of the collection for storing FSM states and data

classmethod from_url(url: str, $connection_kwargs$: $Dict[str, Any] \mid None = None$, **kwargs: Any) \rightarrow MongoStorage

Create an instance of MongoStorage with specifying the connection string

Parameters

- url for example mongodb://user:password@host:port
- connection_kwargs see motor docs
- **kwargs** arguments to be passed to *MongoStorage*

Returns

an instance of MongoStorage

KeyBuilder

Keys inside Redis and Mongo storages can be customized via key builders:

```
class aiogram.fsm.storage.base.KeyBuilder
```

Base class for key builder.

abstract build(*key: StorageKey, part: Literal*['data', 'state', 'lock'] | None = None) \rightarrow str Build key to be used in storage's db queries

Parameters

- **key** contextual key
- part part of the record

Returns

key to be used in storage's db queries

```
class aiogram.fsm.storage.base.DefaultKeyBuilder(*, prefix: str = 'fsm', separator: str = ':',
                                                              with bot id: bool = False,
                                                              with\_business\_connection\_id: bool = False,
                                                              with\_destiny: bool = False)
      Simple key builder with default prefix.
      Generates a colon-joined string with prefix, chat_id, user_id, optional bot_id, business_connection_id, destiny
      and field.
      Format:
           <prefix>:<bot_id?>:<business_connection_id?>:<chat_id>:<user_id>:<destiny?</pre>
           >:<field?>
      build(key: StorageKey, part: Literal['data', 'state', 'lock'] | None = None) \rightarrow str
           Build key to be used in storage's db queries
               Parameters
                    • key – contextual key
                    • part – part of the record
               Returns
                    key to be used in storage's db queries
Writing own storages
class aiogram.fsm.storage.base.BaseStorage
      Base class for all FSM storages
      abstract async set_state(key: StorageKey, state: str \mid State \mid None = None) \rightarrow None
           Set state for specified key
               Parameters
                    • key - storage key
                    • state - new state
      abstract async get_state(key: StorageKey) \rightarrow str | None
           Get key state
               Parameters
                   key - storage key
               Returns
                   current state
      abstract async set_data(key: StorageKey, data: Dict[str, Any]) \rightarrow None
           Write data (replace)
               Parameters
                    • key - storage key
                    • data - new data
      abstract async get_data(key: StorageKey) \rightarrow Dict[str, Any]
```

Get current data for key

Parameters

key – storage key

Returns

current data

async update_data($key: StorageKey, data: Dict[str, Any]) \rightarrow Dict[str, Any]$

Update date in the storage for key (like dict.update)

Parameters

- **key** storage key
- data partial data

Returns

new data

abstract async close() \rightarrow None

Close storage (database connection, file or etc.)

Strategy

This module provides the FSMStrategy enumeration which is used to define the strategy of the finite state machine.

```
class aiogram.fsm.strategy.FSMStrategy(value, names=None, *, module=None, qualname=None,
                                           type=None, start=1, boundary=None)
```

FSM strategy for storage key generation.

CHAT = 2

State will be stored for each chat globally without separating by users.

$CHAT_TOPIC = 5$

State will be stored for each chat and topic, but not separated by users.

$GLOBAL_USER = 3$

State will be stored globally for each user globally.

$USER_IN_CHAT = 1$

State will be stored for each user in chat.

$USER_IN_TOPIC = 4$

State will be stored for each user in chat and topic.

Scenes Wizard

Added in version 3.2.



Warning

This feature is experimental and may be changed in future versions.

aiogram's basics API is easy to use and powerful, allowing the implementation of simple interactions such as triggering a command or message for a response. However, certain tasks require a dialogue between the user and the bot. This is where Scenes come into play.

Understanding Scenes

A Scene in **aiogram** is like an abstract, isolated namespace or room that a user can be ushered into via the code. When a user is within a Scene, most other global commands or message handlers are bypassed, unless they are specifically designed to function outside of the Scenes. This helps in creating an experience of focused interactions. Scenes provide a structure for more complex interactions, effectively isolating and managing contexts for different stages of the conversation. They allow you to control and manage the flow of the conversation in a more organized manner.

Scene Lifecycle

Each Scene can be "entered", "left" or "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

596

For example we have a quiz bot, which asks the user a series of questions and then displays the results.

Lets start with the data models, in this example simple data models are used to represent the questions and answers, in real life you would probably use a database to store the data.

Listing 3: Questions list

```
@dataclass
class Answer:
    """
    Represents an answer to a question.
    """
    text: str
    """The answer text"""
```

```
is_correct: bool = False
    """Indicates if the answer is correct"""
@dataclass
class Question:
    Class representing a quiz with a question and a list of answers.
   text: str
   """The question text"""
   answers: list[Answer]
   """List of answers"""
   correct_answer: str = field(init=False)
   def __post_init__(self):
       self.correct_answer = next(answer.text for answer in self.answers if answer.is_
# Fake data, in real application you should use a database or something else
QUESTIONS = [
   Question(
       text="What is the capital of France?",
        answers=[
            Answer("Paris", is_correct=True),
            Answer("London"),
           Answer("Berlin"),
           Answer("Madrid"),
       ],
   ),
   Question(
       text="What is the capital of Spain?",
        answers=[
            Answer("Paris"),
           Answer("London"),
           Answer("Berlin"),
            Answer("Madrid", is_correct=True),
       ],
   ),
   Question(
       text="What is the capital of Germany?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin", is_correct=True),
            Answer("Madrid"),
       ],
   ),
    Question(
```

```
text="What is the capital of England?",
    answers=[
        Answer("Paris"),
        Answer("London", is_correct=True),
        Answer("Berlin"),
        Answer("Madrid"),
    ],
),
Question(
    text="What is the capital of Italy?",
    answers=[
        Answer("Paris"),
        Answer("London"),
        Answer("Berlin"),
        Answer("Rome", is_correct=True),
    ],
),
```

Then, we need to create a Scene class that will represent the quiz game scene:

1 Note

Keyword argument passed into class definition describes the scene name - is the same as state of the scene.

Listing 4: Quiz Scene

```
class QuizScene(Scene, state="quiz"):
    """
    This class represents a scene for a quiz game.

It inherits from Scene class and is associated with the state "quiz".

It handles the logic and flow of the quiz game.

"""
```

Also we need to define a handler that helps to start the quiz game:

Listing 5: Start command handler

```
quiz_router = Router(name=__name__)
# Add handler that initializes the scene
quiz_router.message.register(QuizScene.as_handler(), Command("quiz"))
```

Once the scene is defined, we need to register it in the SceneRegistry:

Listing 6: Registering the scene

```
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

When all questions are answered, we should show the results to the user, as you can see in the code below, we use *await self.wizard.exit()* to exit from the scene when questions list is over in the *QuizScene.on_enter* handler.

Thats means that we need to write an exit handler to show the results to the user:

Listing 10: Show results handler

```
@on.message.exit()
  async def on_exit(self, message: Message, state: FSMContext) → None:
    """
    Method triggered when the user exits the quiz scene.

It calculates the user's answers, displays the summary, and clears the stored.
    →answers.
```

```
:param message:
:param state:
:return:
data = await state.get_data()
answers = data.get("answers", {})
correct = 0
incorrect = 0
user_answers = []
for step, quiz in enumerate(QUESTIONS):
    answer = answers.get(step)
    is_correct = answer == quiz.correct_answer
    if is_correct:
        correct += 1
        icon = ""
    else:
        incorrect += 1
        icon = ""
    if answer is None:
        answer = "no answer"
    user_answers.append(f"{quiz.text} ({icon} {html.quote(answer)})")
content = as_list(
    as_section(
        Bold("Your answers:"),
        as_numbered_list(*user_answers),
    ),
    as_section(
        Bold("Summary:"),
        as_list(
            as_key_value("Correct", correct),
            as_key_value("Incorrect", incorrect),
        ),
    ),
)
await message.answer(**content.as_kwargs(), reply_markup=ReplyKeyboardRemove())
await state.set_data({})
```

Also we can implement a actions to exit from the quiz game or go back to the previous question:

Listing 11: Exit handler

```
@on.message(F.text == " Exit")
async def exit(self, message: Message) -> None:
    """
Method triggered when the user selects the "Exit" button.

It exits the quiz.
```

```
:param message:
:return:
"""
await self.wizard.exit()
```

Listing 12: Back handler

```
@on.message(F.text == " Back")
async def back(self, message: Message, state: FSMContext) -> None:
    """
    Method triggered when the user selects the "Back" button.

It allows the user to go back to the previous question.

:param message:
:param state:
:return:
"""
    data = await state.get_data()
    step = data["step"]

previous_step = step - 1
if previous_step < 0:
    # In case when the user tries to go back from the first question,
    # we just exit the quiz
    return await self.wizard.exit()
return await self.wizard.back(step=previous_step)</pre>
```

Now we can run the bot and test the quiz game:

Listing 13: Run the bot

```
async def main():
    dp = create_dispatcher()
    bot = Bot(token=TOKEN)
    await dp.start_polling(bot)

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

Complete them all

Listing 14: Quiz Example

```
import asyncio
import logging
from dataclasses import dataclass, field

(continues on next page)
```

```
from os import getenv
from typing import Any
from aiogram import Bot, Dispatcher, F, Router, html
from aiogram.filters import Command
from aiogram.fsm.context import FSMContext
from aiogram.fsm.scene import Scene, SceneRegistry, ScenesManager, on
from aiogram.fsm.storage.memory import SimpleEventIsolation
from aiogram.types import KeyboardButton, Message, ReplyKeyboardRemove
from aiogram.utils.formatting import (
   Bold,
   as_key_value,
   as_list.
    as_numbered_list,
   as_section,
from aiogram.utils.keyboard import ReplyKeyboardBuilder
TOKEN = getenv("BOT_TOKEN")
@dataclass
class Answer:
   Represents an answer to a question.
   text: str
   """The answer text"""
   is_correct: bool = False
    """Indicates if the answer is correct"""
@dataclass
class Question:
    Class representing a quiz with a question and a list of answers.
   text: str
   """The question text"""
   answers: list[Answer]
   """List of answers"""
   correct_answer: str = field(init=False)
   def __post_init__(self):
        self.correct_answer = next(answer.text for answer in self.answers if answer.is_
# Fake data, in real application you should use a database or something else
```

```
QUESTIONS = \Gamma
    Question(
        text="What is the capital of France?",
        answers=[
            Answer("Paris", is_correct=True),
            Answer("London"),
            Answer("Berlin"),
            Answer("Madrid"),
        ],
    ),
    Question(
        text="What is the capital of Spain?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin"),
            Answer("Madrid", is_correct=True),
        ],
    ),
    Question(
        text="What is the capital of Germany?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin", is_correct=True),
            Answer("Madrid"),
        ],
    ),
    Question(
        text="What is the capital of England?",
        answers=[
            Answer("Paris"),
            Answer("London", is_correct=True),
            Answer("Berlin"),
            Answer("Madrid"),
        ],
    ),
    Question(
        text="What is the capital of Italy?",
        answers=[
            Answer("Paris"),
            Answer("London"),
            Answer("Berlin"),
            Answer("Rome", is_correct=True),
        ],
    ),
]
class QuizScene(Scene, state="quiz"):
    This class represents a scene for a quiz game.
```

```
It inherits from Scene class and is associated with the state "quiz".
   It handles the logic and flow of the quiz game.
   @on.message.enter()
   async def on_enter(self, message: Message, state: FSMContext, step: int | None = 0) -
\rightarrow Any:
       Method triggered when the user enters the quiz scene.
       It displays the current question and answer options to the user.
       :param message:
       :param state:
       :param step: Scene argument, can be passed to the scene using the wizard
       if not step:
           # This is the first step, so we should greet the user
           await message.answer("Welcome to the quiz!")
       try:
           quiz = QUESTIONS[step]
       except IndexError:
           # This error means that the question's list is over
           return await self.wizard.exit()
       markup = ReplyKeyboardBuilder()
       markup.add(*[KeyboardButton(text=answer.text) for answer in quiz.answers])
       if step > 0:
           markup.button(text=" Back")
       markup.button(text=" Exit")
       await state.update_data(step=step)
       return await message.answer(
           text=QUESTIONS[step].text,
           reply_markup=markup.adjust(2).as_markup(resize_keyboard=True),
       )
   @on.message.exit()
   async def on_exit(self, message: Message, state: FSMContext) -> None:
       Method triggered when the user exits the quiz scene.
       It calculates the user's answers, displays the summary, and clears the stored.
⊶answers.
       :param message:
       :param state:
       :return:
```

```
......
    data = await state.get_data()
    answers = data.get("answers", {})
    correct = 0
    incorrect = 0
    user_answers = []
    for step, quiz in enumerate(QUESTIONS):
        answer = answers.get(step)
        is_correct = answer == quiz.correct_answer
        if is_correct:
            correct += 1
            icon = ""
        else:
            incorrect += 1
            icon = ""
        if answer is None:
            answer = "no answer"
        user_answers.append(f"{quiz.text} ({icon} {html.quote(answer)})")
    content = as_list(
        as_section(
            Bold("Your answers:"),
            as_numbered_list(*user_answers),
        ),
        as_section(
            Bold("Summary:"),
            as_list(
                as_key_value("Correct", correct),
                as_key_value("Incorrect", incorrect),
            ),
        ),
    )
    await message.answer(**content.as_kwargs(), reply_markup=ReplyKeyboardRemove())
    await state.set_data({})
@on.message(F.text == " Back")
async def back(self, message: Message, state: FSMContext) -> None:
    Method triggered when the user selects the "Back" button.
    It allows the user to go back to the previous question.
    :param message:
    :param state:
    :return:
    data = await state.get_data()
    step = data["step"]
```

(continued from previous page)

```
previous_step = step - 1
       if previous_step < 0:</pre>
           # In case when the user tries to go back from the first question,
           # we just exit the quiz
           return await self.wizard.exit()
       return await self.wizard.back(step=previous_step)
   @on.message(F.text == " Exit")
   async def exit(self, message: Message) -> None:
       Method triggered when the user selects the "Exit" button.
       It exits the quiz.
       :param message:
       :return:
       await self.wizard.exit()
   @on.message(F.text)
   async def answer(self, message: Message, state: FSMContext) -> None:
       Method triggered when the user selects an answer.
       It stores the answer and proceeds to the next question.
       :param message:
       :param state:
        :return:
       data = await state.get_data()
       step = data["step"]
       answers = data.get("answers", {})
       answers[step] = message.text
       await state.update_data(answers=answers)
       await self.wizard.retake(step=step + 1)
   @on.message()
   async def unknown_message(self, message: Message) -> None:
       Method triggered when the user sends a message that is not a command or an.
⇔answer.
       It asks the user to select an answer.
       :param message: The message received from the user.
       :return: None
       await message.answer("Please select an answer.")
```

(continues on next page)

(continued from previous page)

```
quiz_router = Router(name=__name__)
# Add handler that initializes the scene
quiz_router.message.register(QuizScene.as_handler(), Command("quiz"))
@quiz_router.message(Command("start"))
async def command_start(message: Message, scenes: ScenesManager):
   await scenes.close()
   await message.answer(
        "Hi! This is a quiz bot. To start the quiz, use the /quiz command.",
       reply_markup=ReplyKeyboardRemove(),
   )
def create_dispatcher():
    # Event isolation is needed to correctly handle fast user responses
   dispatcher = Dispatcher(
        events_isolation=SimpleEventIsolation(),
   dispatcher.include_router(quiz_router)
   # To use scenes, you should create a SceneRegistry and register your scenes there
   scene_registry = SceneRegistry(dispatcher)
    # ... and then register a scene in the registry
    # by default, Scene will be mounted to the router that passed to the SceneRegistry,
    # but you can specify the router explicitly using the `router` argument
   scene_registry.add(QuizScene)
   return dispatcher
async def main():
   dp = create_dispatcher()
   bot = Bot(token=TOKEN)
    await dp.start_polling(bot)
if __name__ == "__main__":
    # Alternatively, you can use aiogram-cli:
    # `aiogram run polling quiz_scene:create_dispatcher --log-level info --token BOT_
→ TOKEN`
   logging.basicConfig(level=logging.INFO)
    asyncio.run(main())
```

Components

- aiogram.fsm.scene.Scene represents a scene, contains handlers
- aiogram.fsm.scene.SceneRegistry container for all scenes in the bot, used to register scenes and resolve them by name
- aiogram.fsm.scene.ScenesManager manages scenes for each user, used to enter, leave and resolve current scene for user
- aiogram.fsm.scene.SceneConfig scene configuration, used to configure scene
- aiogram.fsm.scene.SceneWizard scene wizard, used to interact with user in scene from active scene handler
- Markers marker for scene handlers, used to mark scene handlers

class aiogram.fsm.scene.Scene(wizard: SceneWizard)

Represents a scene in a conversation flow.

A scene is a specific state in a conversation where certain actions can take place.

Each scene has a set of filters that determine when it should be triggered, and a set of handlers that define the actions to be executed when the scene is active.



This class is not meant to be used directly. Instead, it should be subclassed to define custom scenes.

classmethod add_to_router(router: Router) \rightarrow None

Adds the scene to the given router.

Parameters

router

Returns

classmethod as_handler(**kwargs: Any) → Callable[[...], Any]

Create an entry point handler for the scene, can be used to simplify the handler that starts the scene.

```
>>> router.message.register(MyScene.as_handler(), Command("start"))
```

classmethod as_router($name: str \mid None = None$) $\rightarrow Router$

Returns the scene as a router.

Returns

new router

class aiogram.fsm.scene.**SceneRegistry**(router: Router, register on add: bool = True)

A class that represents a registry for scenes in a Telegram bot.

add(*scenes: Type[Scene], router: Router | None = None) \rightarrow None

This method adds the specified scenes to the registry and optionally registers it to the router.

If a scene with the same state already exists in the registry, a SceneException is raised.

Warning

If the router is not specified, the scenes will not be registered to the router. You will need to include the scenes manually to the router or use the register method.

Parameters

- scenes A variable length parameter that accepts one or more types of scenes. These scenes are instances of the Scene class.
- router An optional parameter that specifies the router to which the scenes should be added.

Returns

None

```
get(scene: Type[Scene] \mid str \mid None) \rightarrow Type[Scene]
```

This method returns the registered Scene object for the specified scene. The scene parameter can be either a Scene object or a string representing the name of the scene. If a Scene object is provided, the state attribute of the SceneConfig object associated with the Scene object will be used as the scene name. If None or an invalid type is provided, a SceneException will be raised.

If the specified scene is not registered in the SceneRegistry object, a SceneException will be raised.

Parameters

scene – A Scene object or a string representing the name of the scene.

Returns

The registered Scene object corresponding to the given scene parameter.

```
register(*scenes: Type[Scene]) \rightarrow None
```

Registers one or more scenes to the SceneRegistry.

Parameters

scenes – One or more scene classes to register.

Returns

None

class aiogram.fsm.scene.**ScenesManager**(registry: SceneRegistry, update_type: str, event: TelegramObject, state: FSMContext, data: Dict[str, Any])

The ScenesManager class is responsible for managing scenes in an application. It provides methods for entering and exiting scenes, as well as retrieving the active scene.

```
async close(**kwargs: Any) \rightarrow None
```

Close method is used to exit the currently active scene in the ScenesManager.

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

Returns

None

async enter($scene_type$: Type[Scene] | str | None, $_check_active$: bool = True, **kwargs: Any) \rightarrow NoneEnters 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

actions: Dict[SceneAction, Dict[str, CallableObject]]

Scene actions

 $attrs_resolver() \rightarrow Generator[tuple[str, Any], None, None]$

Attributes resolver.



This attribute should only be changed when you know what you are doing.

Added in version 3.19.0.

callback_query_without_state: bool | None = None

Allow callback query without state

handlers: List[HandlerContainer]

Scene handlers

reset_data_on_enter: bool | None = None

Reset scene data on enter

reset_history_on_enter: bool | None = None

Reset scene history on enter

state: str | None

Scene state

A class that represents a wizard for managing scenes in a Telegram bot.

Instance of this class is passed to each scene as a parameter. So, you can use it to transition between scenes, get and set data, etc.

1 Note

This class is not meant to be used directly. Instead, it should be used as a parameter in the scene constructor.

```
async back(**kwargs: Any) \rightarrow None
```

This method is used to go back to the previous scene.

Parameters

kwargs – Keyword arguments that can be passed to the method.

Returns

None

$\textbf{async clear_data()} \rightarrow None$

Clears the data.

Returns

None

async enter(**
$$kwargs: Any$$
) \rightarrow None

Enter method is used to transition into a scene in the SceneWizard class. It sets the state, clears data and history if specified, and triggers entering event of the scene.

Parameters

kwargs – Additional keyword arguments.

Returns

None

```
async exit(**kwargs: Any) \rightarrow None
```

Exit the current scene and enter the default scene/state.

Parameters

kwargs – Additional keyword arguments.

Returns

None

```
\textbf{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] | 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 three points for transitions:

- enter point when user enters to the scene
- leave point when user leaves the scene and the enter another scene
- exit point when user exits from the scene

2.4.7 Middlewares

aiogram provides powerful mechanism for customizing event handlers via middlewares.

Middlewares in bot framework seems like Middlewares mechanism in web-frameworks like aiohttp, fastapi, Django or etc.) with small difference - here is implemented two layers of middlewares (before and after filters).



Middleware is function that triggered on every event received from Telegram Bot API in many points on processing pipeline.

Base theory

As many books and other literature in internet says:

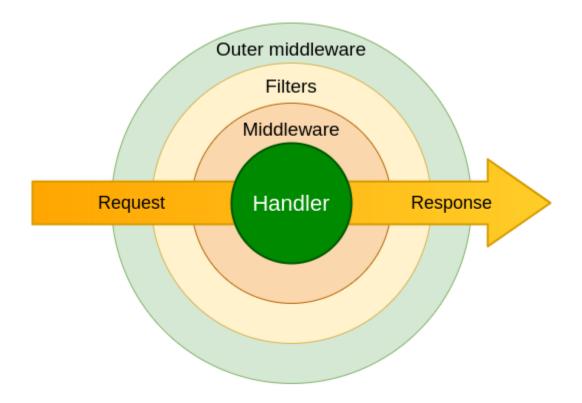
Middleware is reusable software that leverages patterns and frameworks to bridge the gap between the functional requirements of applications and the underlying operating systems, network protocol stacks, and databases.

Middleware can modify, extend or reject processing event in many places of pipeline.

Basics

Middleware instance can be applied for every type of Telegram Event (Update, Message, etc.) in two places

- 1. Outer scope before processing filters (<router>.<event>.outer_middleware(...))
- 2. Inner scope after processing filters but before handler (<router>.<event>.middleware(...))



A Attention

Middleware should be subclass of BaseMiddleware (from aiogram import BaseMiddleware) or any async callable

Arguments specification

class aiogram.dispatcher.middlewares.base.BaseMiddleware

Bases: ABC

Generic middleware class

abstract async __call__(handler: Callable[[TelegramObject, Dict[str, Any]], Awaitable[Any]], event: TelegramObject, data: Dict[str, Any]) \rightarrow Any

Execute middleware

Parameters

• handler – Wrapped handler in middlewares chain

- event Incoming event (Subclass of aiogram.types.base.TelegramObject)
- data Contextual data. Will be mapped to handler arguments

Returns

Any

Examples



Middleware should always call await handler(event, data) to propagate event for next middleware/handler. If you want to stop processing event in middleware you should not call await handler(event, data).

Class-based

```
from aiogram import BaseMiddleware
from aiogram.types import Message

class CounterMiddleware(BaseMiddleware):
    def __init__(self) -> None:
        self.counter = 0

    async def __call__(
        self,
        handler: Callable[[Message, Dict[str, Any]], Awaitable[Any]],
        event: Message,
        data: Dict[str, Any]
) -> Any:
        self.counter += 1
        data['counter'] = self.counter
        return await handler(event, data)
```

and then

```
router = Router()
router.message.middleware(CounterMiddleware())
```

Function-based

```
@dispatcher.update.outer_middleware()
async def database_transaction_middleware(
    handler: Callable[[Update, Dict[str, Any]], Awaitable[Any]],
    event: Update,
    data: Dict[str, Any]
) -> Any:
    async with database.transaction():
        return await handler(event, data)
```

Facts

- 1. Middlewares from outer scope will be called on every incoming event
- 2. Middlewares from inner scope will be called only when filters pass
- 3. Inner middlewares is always calls for aiogram.types.update.Update event type in due to all incoming updates going to specific event type handler through built in update handler

2.4.8 Errors

Handling errors

Is recommended way that you should use errors inside handlers using try-except block, but in common cases you can use global errors handler at router or dispatcher level.

If you specify errors handler for router - it will be used for all handlers inside this router.

If you specify errors handler for dispatcher - it will be used for all handlers inside all routers.

```
@router.error(ExceptionTypeFilter(MyCustomException), F.update.message.as_("message"))
async def handle_my_custom_exception(event: ErrorEvent, message: Message):
    # do something with error
    await message.answer("Oops, something went wrong!")

@router.error()
async def error_handler(event: ErrorEvent):
    logger.critical("Critical error caused by %s", event.exception, exc_info=True)
    # do something with error
    ...
```

ErrorEvent

Internal event, should be used to receive errors while processing Updates from Telegram

Source: https://core.telegram.org/bots/api#error-event

update: Update
 Received update
exception: Exception

Exception

Error types

exception aiogram.exceptions.AiogramError

Base exception for all aiogram errors.

exception aiogram.exceptions.DetailedAiogramError(message: str)

Base exception for all aiogram errors with detailed message.

exception aiogram.exceptions.CallbackAnswerException

Exception for callback answer.

exception aiogram.exceptions.SceneException

Exception for scenes.

exception aiogram.exceptions.UnsupportedKeywordArgument(message: str)

Exception raised when a keyword argument is passed as filter.

exception aiogram.exceptions.**TelegramAPIError**(method: TelegramMethod, message: str)

Base exception for all Telegram API errors.

exception aiogram.exceptions.**TelegramNetworkError**(method: TelegramMethod, message: str)

Base exception for all Telegram network errors.

exception 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.

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

Exception raised when chat, message, user, etc. not found.

exception aiogram.exceptions.**TelegramConflictError**(method: TelegramMethod, message: str)

Exception raised when bot token is already used by another application in polling mode.

exception aiogram.exceptions.**TelegramUnauthorizedError**(method: TelegramMethod, message: str)

Exception raised when bot token is invalid.

exception aiogram.exceptions.**TelegramForbiddenError**(method: TelegramMethod, message: str)

Exception raised when bot is kicked from chat or etc.

exception aiogram.exceptions.TelegramServerError(method: TelegramMethod, message: str)

Exception raised when Telegram server returns 5xx error.

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

Exception raised when Telegram server is restarting.

It seems like this error is not used by Telegram anymore, but it's still here for backward compatibility.

Currently, you should expect that Telegram can raise RetryAfter (with timeout 5 seconds)

error instead of this one.

exception aiogram.exceptions.**TelegramEntityTooLarge**(*method: TelegramMethod, message: str*) Exception raised when you are trying to send a file that is too large.

exception aiogram.exceptions.**ClientDecodeError**(*message: str, original: Exception, data: Any*) Exception raised when client can't decode response. (Malformed response, etc.)

2.4.9 Flags

Flags is a markers for handlers that can be used in *middlewares* or special *utilities* to make classification of the handlers. Flags can be added to the handler via *decorators*, *handlers registration* or filters.

Via decorators

For example mark handler with chat_action flag

```
from aiogram import flags

@flags.chat_action
async def my_handler(message: Message)
```

Or just for rate-limit or something else

```
from aiogram import flags

@flags.rate_limit(rate=2, key="something")
async def my_handler(message: Message)
```

Via handler registration method

```
@router.message(..., flags={'chat_action': 'typing', 'rate_limit': {'rate': 5}})
```

Via filters

```
class Command(Filter):
    ...

def update_handler_flags(self, flags: Dict[str, Any]) -> None:
    commands = flags.setdefault("commands", [])
    commands.append(self)
```

Use in middlewares

 $aiogram.dispatcher.flags.\textbf{check_flags}(\textit{handler: HandlerObject} \mid \textit{Dict[str, Any], magic: MagicFilter)} \rightarrow \\ Any$

Check flags via magic filter

Parameters

- handler handler object or data
- magic instance of the magic

Returns

the result of magic filter check

 $\label{eq:aiogram.dispatcher.flags.extract_flags} (\textit{handler: HandlerObject} \mid \textit{Dict[str, Any]}) \rightarrow \textit{Dict[str, Any]}) \rightarrow \textit{Dict[str, Any]}$ Extract flags from handler or middleware context data

Parameters

handler – handler object or data

Returns

dictionary with all handler flags

```
aiogram.dispatcher.flags.get_flag(handler: HandlerObject | Dict[str, Any], name: str, *, default: Any | None = None) \rightarrow Any
```

Get flag by name

Parameters

- handler handler object or data
- name name of the flag
- **default** default value (None)

Returns

value of the flag or default

Example in middlewares

```
async def my_middleware(handler, event, data):
    typing = get_flag(data, "typing") # Check that handler marked with `typing` flag
    if not typing:
        return await handler(event, data)

async with ChatActionSender.typing(chat_id=event.chat.id):
        return await handler(event, data)
```

Use in utilities

For example you can collect all registered commands with handler description and then it can be used for generating commands help

```
def collect_commands(router: Router) -> Generator[Tuple[Command, str], None, None]:
   for handler in router.message.handlers:
       if "commands" not in handler.flags: # ignore all handler without commands
            continue
        # the Command filter adds the flag with list of commands attached to the handler
       for command in handler.flags["commands"]:
            yield command, handler.callback.__doc__ or ""
   # Recursively extract commands from nested routers
   for sub_router in router.sub_routers:
       yield from collect_commands(sub_router)
```

2.4.10 Webhook

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



If you use webhook, you can't use long polling at the same time.

Before start i'll recommend you to read official Telegram's documentation about webhook

After you read it, you can start to read this section.

Generally to use webhook with aiogram you should use any async web framework. By out of the box aiogram has an aiohttp integration, so we'll use it.



1 Note

You can use any async web framework you want, but you should write your own integration if you don't use aiohttp.

aiohttp integration

Out of the box aiogram has aiohttp integration, so you can use it.

Here is available few ways to do it using different implementations of the webhook controller:

- aiogram.webhook.aiohttp_server.BaseRequestHandler Abstract class for aiohttp webhook controller
- $\bullet \ \ aiogram. we bhook. aiohttp_server. Simple Request Handler Simple we bhook controller, uses single Bot$
- aiogram.webhook.aiohttp_server.TokenBasedRequestHandler-Token based webhook controller, uses multiple Bot instances and tokens

You can use it as is or inherit from it and override some methods.

__init__(*dispatcher*: Dispatcher, *handle_in_background*: bool = False, **data: Any) \rightarrow None Base handler that helps to handle incoming request from an aiohttp and propagate it to the Dispatcher

Parameters

- dispatcher instance of aiogram.dispatcher.dispatcher.Dispatcher
- handle_in_background immediately responds to the Telegram instead of a waiting end of a handler process

```
register(app: None, /, path: str, **kwargs: Any) \rightarrow None Register route and shutdown callback
```

Parameters

- app instance of aiohttp Application
- **path** route path
- kwargs

```
abstract async resolve\_bot(request: Request) \rightarrow Bot
```

This method should be implemented in subclasses of this class.

Resolve Bot instance from request.

Parameters request

Returns

Bot instance

```
class aiogram.webhook.aiohttp_server.SimpleRequestHandler(dispatcher: Dispatcher, bot: Bot, handle_in_background: bool = True, secret_token: str \mid None = None, **data: Any)
```

```
__init__(dispatcher: Dispatcher, bot: Bot, handle_in_background: bool = True, secret_token: str \mid None = None, **data: Any) \rightarrow None
```

Handler for single Bot instance

Parameters

- dispatcher instance of aiogram.dispatcher.dispatcher.Dispatcher
- handle_in_background immediately responds to the Telegram instead of a waiting end of handler process
- **bot** instance of aiogram.client.bot.Bot

```
async close() \rightarrow None
```

Close bot session

```
register(app: None, /, path: str, **kwargs: Any) → None
```

Register route and shutdown callback

Parameters

• app – instance of aiohttp Application

- path route path
- kwargs

```
async resolve_bot(request: Request) \rightarrow Bot
```

This method should be implemented in subclasses of this class.

Resolve Bot instance from request.

Parameters

request

Returns

Bot instance

class aiogram.webhook.aiohttp_server.TokenBasedRequestHandler(dispatcher: Dispatcher,

handle_in_background: bool = True, bot_settings: Dict[str, Any] | None = None, **data: Any)

__init__(dispatcher: Dispatcher, handle_in_background: bool = True, bot_settings: Dict[str, Any] | None = *None*, ***data: Any*) \rightarrow None

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



1 Note

This handler is not recommended in due to token is available in URL and can be logged by reverse proxy server or other middleware.

Parameters

- dispatcher instance of aiogram.dispatcher.dispatcher.Dispatcher
- handle_in_background immediately responds to the Telegram instead of a waiting end of handler process
- bot_settings kwargs that will be passed to new Bot instance

register($app: None, /, path: str, **kwargs: Any) \rightarrow None$

Validate path, register route and shutdown callback

Parameters

- app instance of aiohttp Application
- path route path
- kwargs

async resolve_bot(request: Request) \rightarrow Bot

Get bot token from a path and create or get from cache Bot instance

Parameters

request

Returns

Security

Telegram supports two methods to verify incoming requests that they are from Telegram:

Using a secret token

When you set webhook, you can specify a secret token and then use it to verify incoming requests.

Using IP filtering

You can specify a list of IP addresses from which you expect incoming requests, and then use it to verify incoming requests.

It can be acy using firewall rules or nginx configuration or middleware on application level.

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

```
\label{lem:aiogram.webhook.aiohttp_server.ip_filter_middleware} (\textit{ip\_filter: IPFilter}) \rightarrow Callable[[Request, Callable[[Request], Awaitable[StreamResponse]]], Awaitable[Any]]
```

```
Parameters
ip_filter
```

Returns

```
class aiogram.webhook.security.IPFilter(ips: Sequence[str | IPv4Network | IPv4Address] | None = None)
    __init__(ips: Sequence[str | IPv4Network | IPv4Address] | None = None)
```

Examples

Behind reverse proxy

In this example we'll use aiohttp as web framework and nginx as reverse proxy.

```
This example shows how to use webhook on behind of any reverse proxy (nginx, traefik, ingress etc.)

import logging import sys from os import getenv

from aiohttp import web

from aiogram import Bot, Dispatcher, Router from aiogram.client.default import DefaultBotProperties from aiogram.enums import ParseMode from aiogram.filters import CommandStart from aiogram.types import Message
```

(continues on next page)

(continued from previous page)

```
from aiogram.utils.markdown import hbold
from aiogram.webhook.aiohttp_server import SimpleRequestHandler, setup_application
# Bot token can be obtained via https://t.me/BotFather
TOKEN = getenv("BOT_TOKEN")
# Webserver settings
# bind localhost only to prevent any external access
WEB\_SERVER\_HOST = "127.0.0.1"
# Port for incoming request from reverse proxy. Should be any available port
WEB\_SERVER\_PORT = 8080
# Path to webhook route, on which Telegram will send requests
WEBHOOK_PATH = "/webhook"
# Secret key to validate requests from Telegram (optional)
WEBHOOK_SECRET = "my-secret"
# Base URL for webhook will be used to generate webhook URL for Telegram,
# in this example it is used public DNS with HTTPS support
BASE_WEBHOOK_URL = "https://aiogram.dev"
# All handlers should be attached to the Router (or Dispatcher)
router = Router()
@router.message(CommandStart())
async def command_start_handler(message: Message) -> None:
    This handler receives messages with `/start` command
    # Most event objects have aliases for API methods that can be called in events'
    # For example if you want to answer to incoming message you can use `message.answer(.
→...)` alias
    # and the target chat will be passed to :ref:`aiogram.methods.send_message.
→ SendMessage`
    # method automatically or call API method directly via
    # Bot instance: `bot.send_message(chat_id=message.chat.id, ...)`
    await message.answer(f"Hello, {hbold(message.from_user.full_name)}!")
@router.message()
async def echo_handler(message: Message) -> None:
    Handler will forward receive a message back to the sender
   By default, message handler will handle all message types (like text, photo, sticker.
⇔etc.)
   try:
        # Send a copy of the received message
        await message.send_copy(chat_id=message.chat.id)
    except TypeError:
```

(continues on next page)

(continued from previous page)

```
# But not all the types is supported to be copied so need to handle it
        await message.answer("Nice try!")
async def on_startup(bot: Bot) -> None:
    # If you have a self-signed SSL certificate, then you will need to send a public
    # certificate to Telegram
   await bot.set_webhook(f"{BASE_WEBHOOK_URL}{WEBHOOK_PATH}", secret_token=WEBHOOK_
→SECRET)
def main() -> None:
    # Dispatcher is a root router
   dp = Dispatcher()
    # ... and all other routers should be attached to Dispatcher
   dp.include_router(router)
    # Register startup hook to initialize webhook
   dp.startup.register(on_startup)
    # Initialize Bot instance with default bot properties which will be passed to all
→API calls
   bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))
   # Create aiohttp.web.Application instance
   app = web.Application()
   # Create an instance of request handler.
    # aiogram has few implementations for different cases of usage
    # In this example we use SimpleRequestHandler which is designed to handle simple.
   webhook_requests_handler = SimpleRequestHandler(
        dispatcher=dp.
       bot=bot.
        secret_token=WEBHOOK_SECRET,
    # Register webhook handler on application
   webhook_requests_handler.register(app, path=WEBHOOK_PATH)
   # Mount dispatcher startup and shutdown hooks to aiohttp application
   setup_application(app, dp, bot=bot)
    # And finally start webserver
   web.run_app(app, host=WEB_SERVER_HOST, port=WEB_SERVER_PORT)
if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, stream=sys.stdout)
   main()
```

When you use nginx as reverse proxy, you should set *proxy_pass* to your aiohttp server address.

```
location /webhook {
    proxy_set_header Host $http_host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_redirect off;
    proxy_buffering off;
    proxy_pass http://127.0.0.1:8080;
}
```

Without reverse proxy (not recommended)

In case without using reverse proxy, you can use aiohttp's ssl context.

Also this example contains usage with self-signed certificate.

```
This example shows how to use webhook with SSL certificate.
import logging
import ssl
import sys
from os import getenv
from aiohttp import web
from aiogram import Bot, Dispatcher, Router
from aiogram.client.default import DefaultBotProperties
from aiogram.enums import ParseMode
from aiogram.filters import CommandStart
from aiogram.types import FSInputFile, Message
from aiogram.utils.markdown import hbold
from aiogram.webhook.aiohttp_server import SimpleRequestHandler, setup_application
# Bot token can be obtained via https://t.me/BotFather
TOKEN = getenv("BOT_TOKEN")
# Webserver settings
# bind localhost only to prevent any external access
WEB\_SERVER\_HOST = "127.0.0.1"
# Port for incoming request from reverse proxy. Should be any available port
WEB_SERVER_PORT = 8080
# Path to webhook route, on which Telegram will send requests
WEBHOOK_PATH = "/webhook"
# Secret key to validate requests from Telegram (optional)
WEBHOOK_SECRET = "my-secret"
# Base URL for webhook will be used to generate webhook URL for Telegram,
# in this example it is used public address with TLS support
BASE_WEBHOOK_URL = "https://aiogram.dev"
# Path to SSL certificate and private key for self-signed certificate.
WEBHOOK_SSL_CERT = "/path/to/cert.pem"
```

(continues on next page)

(continued from previous page)

```
WEBHOOK_SSL_PRIV = "/path/to/private.key"
# All handlers should be attached to the Router (or Dispatcher)
router = Router()
@router.message(CommandStart())
async def command_start_handler(message: Message) -> None:
    This handler receives messages with `/start` command
    # Most event objects have aliases for API methods that can be called in events'.
# For example if you want to answer to incoming message you can use `message.answer(.
→...)` alias
   # and the target chat will be passed to :ref:`aiogram.methods.send_message.
→ SendMessage`
    # method automatically or call API method directly via
    # Bot instance: `bot.send_message(chat_id=message.chat.id, ...)`
    await message.answer(f"Hello, {hbold(message.from_user.full_name)}!")
@router.message()
async def echo_handler(message: Message) -> None:
   Handler will forward receive a message back to the sender
   By default, message handler will handle all message types (like text, photo, sticker.
⇔etc.)
    1111111
   try:
        # Send a copy of the received message
       await message.send_copy(chat_id=message.chat.id)
   except TypeError:
        # But not all the types is supported to be copied so need to handle it
        await message.answer("Nice try!")
async def on_startup(bot: Bot) -> None:
    # In case when you have a self-signed SSL certificate, you need to send the.
→certificate
    # itself to Telegram servers for validation purposes
    # (see https://core.telegram.org/bots/self-signed)
    # But if you have a valid SSL certificate, you SHOULD NOT send it to Telegram.
⇔servers.
    await bot.set_webhook(
        f"{BASE_WEBHOOK_URL}{WEBHOOK_PATH}",
        certificate=FSInputFile(WEBHOOK_SSL_CERT),
        secret_token=WEBHOOK_SECRET,
   )
```

(continues on next page)

(continued from previous page)

```
def main() -> None:
    # Dispatcher is a root router
   dp = Dispatcher()
    # ... and all other routers should be attached to Dispatcher
   dp.include_router(router)
    # Register startup hook to initialize webhook
   dp.startup.register(on_startup)
    # Initialize Bot instance with default bot properties which will be passed to all.
→ API calls
   bot = Bot(token=TOKEN, default=DefaultBotProperties(parse_mode=ParseMode.HTML))
    # Create aiohttp.web.Application instance
   app = web.Application()
   # Create an instance of request handler,
    # aiogram has few implementations for different cases of usage
    # In this example we use SimpleRequestHandler which is designed to handle simple.
   webhook_requests_handler = SimpleRequestHandler(
        dispatcher=dp,
       bot=bot,
       secret_token=WEBHOOK_SECRET,
    # Register webhook handler on application
   webhook_requests_handler.register(app, path=WEBHOOK_PATH)
    # Mount dispatcher startup and shutdown hooks to aiohttp application
   setup_application(app, dp, bot=bot)
    # Generate SSL context
    context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
   context.load_cert_chain(WEBHOOK_SSL_CERT, WEBHOOK_SSL_PRIV)
    # And finally start webserver
   web.run_app(app, host=WEB_SERVER_HOST, port=WEB_SERVER_PORT, ssl_context=context)
if __name__ == "__main__":
   logging.basicConfig(level=logging.INFO, stream=sys.stdout)
   main()
```

With using other web framework

You can pass incoming request to aiogram's webhook controller from any web framework you want.

Read more about it in aiogram.dispatcher.dispatcher.Dispatcher.feed_webhook_update() or aiogram. dispatcher.dispatcher.Dispatcher.feed_update() methods.

```
update = Update.model_validate(await request.json(), context={"bot": bot})
await dispatcher.feed_update(update)
```



1 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.

2.4.11 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[str, Any]

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

```
\textbf{class} \  \, \textbf{aiogram.handlers.callback\_query.CallbackQueryHandler}(\textit{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 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.

2.5. Utils 635

Usage example

For example you want to generate inline keyboard with 10 buttons

```
builder = InlineKeyboardBuilder()
for index in range(1, 11):
   builder.button(text=f"Set {index}", callback_data=f"set:{index}")
```

then adjust this buttons to some grid, for example first line will have 3 buttons, the next lines will have 2 buttons

```
builder.adjust(3, 2)
```

also you can attach another builder to this one

```
another_builder = InlineKeyboardBuilder(...)... # Another builder with some buttons
builder.attach(another_builder)
```

or you can attach some already generated markup

```
markup = InlineKeyboardMarkup(inline_keyboard=[...]) # Some markup
builder.attach(InlineKeyboardBuilder.from_markup(markup))
```

and finally you can export this markup to use it in your message

```
await message.answer("Some text here", reply_markup=builder.as_markup())
```

Reply keyboard builder has the same interface

Warning

Note that you can't attach reply keyboard builder to inline keyboard builder and vice versa

Inline Keyboard

class aiogram.utils.keyboard.InlineKeyboardBuilder(markup: List[List[InlineKeyboardButton]] | None = None)

Inline keyboard builder inherits all methods from generic builder

```
button(text: str, url: str | None = None, login_url: LoginUrl | None = None, callback_data: str |
         CallbackData | None = None, switch_inline_query: str | None = None,
         switch_inline_query_current_chat: str | None = None, callback_game: CallbackGame | None =
         None, pay: bool | None = None, **kwargs: Any) \rightarrow aiogram.utils.keyboard.InlineKeyboardBuilder
```

Add new inline button to markup

```
as_markup() → aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup
```

Construct an InlineKeyboardMarkup

```
__init__(markup: List[List[InlineKeyboardButton]] | None = None) <math>\rightarrow None
```

add(**buttons: ButtonType*) → KeyboardBuilder[ButtonType]

Add one or many buttons to markup.

Parameters buttons

Returns

```
adjust(*sizes: int, repeat: bool = False) \rightarrow KeyboardBuilder[ButtonType]
```

Adjust previously added buttons to specific row sizes.

By default, when the sum of passed sizes is lower than buttons count the last one size will be used for tail of the markup. If repeat=True is passed - all sizes will be cycled when available more buttons count than all sizes

Parameters

- sizes
- repeat

Returns

```
property buttons: Generator[ButtonType, None, None]
```

Get flatten set of all buttons

Returns

```
copy() \rightarrow InlineKeyboardBuilder
```

Make full copy of current builder with markup

Returns

```
export() → List[List[ButtonType]]
```

Export configured markup as list of lists of buttons

```
>>> builder = KeyboardBuilder(button_type=InlineKeyboardButton)
>>> ... # Add buttons to builder
>>> markup = InlineKeyboardMarkup(inline_keyboard=builder.export())
```

Returns

classmethod from_markup(markup: InlineKeyboardMarkup) \rightarrow InlineKeyboardBuilder

Create builder from existing markup

Parameters markup

Returns

```
row(*buttons: ButtonType, width: int | None = None) → KeyboardBuilder[ButtonType]
```

Add row to markup

When too much buttons is passed it will be separated to many rows

Parameters

- buttons
- width

Returns

2.5. Utils 637

Reply Keyboard

```
class aiogram.utils.keyboard.ReplyKeyboardBuilder(markup: List[KeyboardButton]] | None =
     Reply keyboard builder inherits all methods from generic builder
     button(text: str, request_contact: bool | None = None, request_location: bool | None = None, request_poll:
              KeyboardButtonPollType | None = None, **kwargs: Any) \rightarrow
              aiogram.utils.keyboard.ReplyKeyboardBuilder
           Add new button to markup
     as_markup() → aiogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
           Construct an ReplyKeyboardMarkup
     __init__(markup: List[List[KeyboardButton]] | None = None) <math>\rightarrow None
     add(*buttons: ButtonType) → KeyboardBuilder[ButtonType]
           Add one or many buttons to markup.
               Parameters
                   buttons
               Returns
     adjust(*sizes: int, repeat: bool = False) \rightarrow KeyboardBuilder[ButtonType]
           Adjust previously added buttons to specific row sizes.
           By default, when the sum of passed sizes is lower than buttons count the last one size will be used for tail
           of the markup. If repeat=True is passed - all sizes will be cycled when available more buttons count than
           all sizes
               Parameters
                   sizes

    repeat

               Returns
     property buttons: Generator[ButtonType, None, None]
           Get flatten set of all buttons
               Returns
     copy() \rightarrow ReplyKeyboardBuilder
           Make full copy of current builder with markup
               Returns
     export() → List[List[ButtonType]]
           Export configured markup as list of lists of buttons
           >>> builder = KeyboardBuilder(button_type=InlineKeyboardButton)
           >>> ... # Add buttons to builder
           >>> markup = InlineKeyboardMarkup(inline_keyboard=builder.export())
```

Returns

```
classmethod from_markup(markup: ReplyKeyboardMarkup) → ReplyKeyboardBuilder
```

Create builder from existing markup

Parameters markup

Returns

row(*buttons: ButtonType, width: int | None = None) \rightarrow KeyboardBuilder[ButtonType] Add row to markup

When too much buttons is passed it will be separated to many rows

Parameters

- buttons
- width

Returns

2.5.2 Translation

In order to make you bot translatable you have to add a minimal number of hooks to your Python code.

These hooks are called translation strings.

The aiogram translation utils is build on top of GNU gettext Python module and Babel library.

Installation

Babel is required to make simple way to extract translation strings from your code

Can be installed from pip directly:

```
pip install Babel
```

or as aiogram extra dependency:

```
pip install aiogram[i18n]
```

Make messages translatable

In order to gettext need to know what the strings should be translated you will need to write translation strings.

For example:

2.5. Utils 639

Danger

f-strings can't be used as translations string because any dynamic variables should be added to message after getting translated message

Also if you want to use translated string in keyword- or magic- filters you will need to use lazy gettext calls:

```
from aiogram import F
from aiogram.utils.i18n import lazy_gettext as __

@router.message(F.text == __("My menu entry"))
...
```

Danger

Lazy gettext calls should always be used when the current language is not know at the moment

\dagger Danger

Lazy gettext can't be used as value for API methods or any Telegram Object (like aiogram.types.inline_keyboard_button.InlineKeyboardButton or etc.)

Working with plural forms

The *gettext* from *aiogram.utils.i18n* is the one alias for two functions _gettext_ and _ngettext_ of GNU gettext Python module. Therefore, the wrapper for message strings is the same _(). You need to pass three parameters to the function: a singular string, a plural string, and a value.

Configuring engine

After you messages is already done to use gettext your bot should know how to detect user language

On top of your application the instance of aiogram.utils.i18n.I18n should be created

```
i18n = I18n(path="locales", default_locale="en", domain="messages")
```

After that you will need to choose one of builtin I18n middleware or write your own.

Builtin middlewares:

Simplel18nMiddleware

Simple I18n middleware.

Chooses language code from the User object received in event

```
__init__(i18n: i18n, i18n_key: str \mid None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None Create an instance of middleware
```

Parameters

- **i18n** instance of I18n
- i18n_key context key for I18n instance
- middleware_key context key for this middleware

Constl18nMiddleware

Const middleware chooses statically defined locale

```
\__init\_(locale: str, i18n: I18n, i18n_key: str | None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None
```

Create an instance of middleware

Parameters

- **i18n** instance of I18n
- i18n_key context key for I18n instance
- middleware_key context key for this middleware

FSMI18nMiddleware

```
class aiogram.utils.i18n.middleware.FSMI18nMiddleware(i18n: i18n, key: str = 'locale', i18n_key: str = None = 'i18n', middleware_key: str = 'i18n_middleware')
```

This middleware stores locale in the FSM storage

```
__init__(i18n: i18n, key: str = 'locale', i18n_key: str | None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None
```

Create an instance of middleware

Parameters

- **i18n** instance of I18n
- **i18n_key** context key for I18n instance
- middleware_key context key for this middleware

async set_locale(state: FSMContext, locale: str) \rightarrow None

Write new locale to the storage

Parameters

- state instance of FSMContext
- locale new locale

2.5. Utils 641

I18nMiddleware

or define you own based on abstract I18nMiddleware middleware:

Abstract I18n middleware.

```
__init__(i18n: i18n, i18n_key: str \mid None = 'i18n', middleware_key: str = 'i18n_middleware') \rightarrow None Create an instance of middleware
```

Parameters

- **i18n** instance of I18n
- i18n_key context key for I18n instance
- middleware_key context key for this middleware

abstract async get_locale(event: TelegramObject, data: Dict[str, Any]) \rightarrow str

Detect current user locale based on event and context.

This method must be defined in child classes

Parameters

- event
- data

Returns

setup(*router*: Router, *exclude*: $Set[str] \mid None = None$) $\rightarrow BaseMiddleware$ Register middleware for all events in the Router

Parameters

- router
- exclude

Returns

Deal with Babel

Step 1 Extract messages

```
pybabel extract --input-dirs=. -o locales/messages.pot
```

Here is --input-dirs=. - path to code and the locales/messages.pot is template where messages will be extracted and *messages* is translation domain.

Working with plural forms

Extracting with Pybabel all strings options:

- -k _:1,1t -k _:1,2 for both singular and plural
- -k __ for lazy strings

```
pybabel extract -k _:1,1t -k _:1,2 -k __ --input-dirs=. -o locales/messages.pot
```

1 Note

Some useful options:

- Add comments for translators, you can use another tag if you want (TR) --add-comments=NOTE
- Contact email for bugreport --msgid-bugs-address=EMAIL
- Disable comments with string location in code --no-location
- Copyrights --copyright-holder=AUTHOR
- Set project name --project=MySuperBot
- Set version --version=2.2

Step 2: Init language

```
pybabel init -i locales/messages.pot -d locales -D messages -l en
```

- -i locales/messages.pot pre-generated template
- -d locales translations directory
- -D messages translations domain
- -1 en language. Can be changed to any other valid language code (For example -1 uk for ukrainian language)

Step 3: Translate texts

To open .po file you can use basic text editor or any PO editor, e.g. Poedit

Just open the file named locales/{language}/LC_MESSAGES/messages.po and write translations

Step 4: Compile translations

```
pybabel compile -d locales -D messages
```

Step 5: Updating messages

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

- Step 5.1: regenerate pot file: command from step 1
- Step 5.2: update po files

```
pybabel \ update \ -d \ locales \ -D \ messages \ -i \ locales/messages.pot
```

- Step 5.3: update your translations: location and tools you know from step 3
- Step 5.4: compile mo files: command from step 4

2.5.3 Chat action sender

Sender

This utility helps to automatically send chat action until long actions is done to take acknowledge bot users the bot is doing something and not crashed.

Provides simply to use context manager.

Technically sender start background task with infinity loop which works until action will be finished and sends the chat action every 5 seconds.

__init__(*, bot: Bot, chat_id: $str \mid int$, $message_thread_id$: $int \mid None = None$, action: str = 'typing', interval: float = 5.0, initial sleep: float = 0.0) \rightarrow None

Parameters

- **bot** instance of the bot
- chat_id target chat id
- message_thread_id unique identifier for the target message thread; supergroups only
- action chat action type
- **interval** interval between iterations
- initial_sleep sleep before first sending of the action

classmethod choose_sticker($chat_id$: $int \mid str$, bot: Bot, $message_thread_id$: $int \mid None = None$, interval: float = 5.0, $initial_sleep$: float = 0.0) $\rightarrow ChatActionSender$

Create instance of the sender with choose_sticker action

classmethod find_location($chat_id$: $int \mid str, bot$: $Bot, message_thread_id$: $int \mid None = None, interval$: $float = 5.0, initial_sleep$: float = 0.0) $\rightarrow ChatActionSender$

Create instance of the sender with find location action

classmethod record_video(chat_id: int | str, bot: Bot, message_thread_id: int | None = None, interval: float = 5.0, initial_sleep: float = 0.0) \rightarrow ChatActionSender

Create instance of the sender with record_video action

classmethod record_video_note($chat_id: int \mid str, bot: Bot, message_thread_id: int \mid None = None, interval: float = 5.0, initial_sleep: float = 0.0) <math>\rightarrow ChatActionSender$

Create instance of the sender with record_video_note action

classmethod record_voice(chat_id: int | str, bot: Bot, message_thread_id: int | None = None, interval: float = 5.0, initial_sleep: float = 0.0) \rightarrow ChatActionSender

Create instance of the sender with record voice action

classmethod typing($chat_id$: $int \mid str, bot$: $Bot, message_thread_id$: $int \mid None = None, interval$: $float = 5.0, initial_sleep$: float = 0.0) $\rightarrow ChatActionSender$

Create instance of the sender with typing action

classmethod upload_document(chat_id: int | str, bot: Bot, message_thread_id: int | None = None, interval: float = 5.0, initial_sleep: float = 0.0) \rightarrow ChatActionSender

Create instance of the sender with *upload_document* action

```
classmethod upload_photo(chat\_id: int \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($chat_id$: $int \mid str, bot$: $Bot, message_thread_id$: $int \mid None = None,$ interval: $float = 5.0, initial_sleep$: float = 0.0) $\rightarrow ChatActionSender$

Create instance of the sender with upload_video_note action

classmethod upload_voice($chat_id$: $int \mid str$, bot: Bot, $message_thread_id$: $int \mid None = None$, interval: float = 5.0, $initial_sleep$: float = 0.0) $\rightarrow ChatActionSender$

Create instance of the sender with *upload_voice* action

Usage

```
async with ChatActionSender.typing(bot=bot, chat_id=message.chat.id):
    # Do something...
# Perform some long calculations
await message.answer(result)
```

Middleware

class aiogram.utils.chat_action.ChatActionMiddleware

Helps to automatically use chat action sender for all message handlers

Usage

Before us a should be registered for the message event

```
<router or dispatcher>.message.middleware(ChatActionMiddleware())
```

After this action all handlers which works longer than *initial_sleep* will produce the 'typing' chat action.

Also sender can be customized via flags feature for particular handler.

Change only action type:

```
@router.message(...)
@flags.chat_action("sticker")
async def my_handler(message: Message): ...
```

Change sender configuration:

```
@router.message(...)
@flags.chat_action(initial_sleep=2, action="upload_document", interval=3)
async def my_handler(message: Message): ...
```

2.5.4 WebApp

Telegram Bot API 6.0 announces a revolution in the development of chatbots using WebApp feature.

You can read more details on it in the official blog and documentation.

aiogram implements simple utils to remove headache with the data validation from Telegram WebApp on the backend side.

Usage

For example from frontend you will pass application/x-www-form-urlencoded POST request with _auth field in body and wants to return User info inside response as application/json

```
from aiogram.utils.web_app import safe_parse_webapp_init_data
from aiohttp.web_request import Request
from aiohttp.web_response import json_response

async def check_data_handler(request: Request):
   bot: Bot = request.app["bot"]

   data = await request.post() # application/x-www-form-urlencoded
   try:
      data = safe_parse_webapp_init_data(token=bot.token, init_data=data["_auth"])
   except ValueError:
      return json_response({"ok": False, "err": "Unauthorized"}, status=401)
      return json_response({"ok": True, "data": data.user.dict()})
```

Functions

```
aiogram.utils.web_app.check_webapp_signature(token: str, init_data: str) → bool
```

Check incoming WebApp init data signature

Source: https://core.telegram.org/bots/webapps#validating-data-received-via-the-web-app

Parameters

- token bot Token
- init_data data from frontend to be validated

Returns

```
aiogram.utils.web_app.parse_webapp_init_data(init_data: str, *, loads: ~typing.Callable[[...], \simtyping.Any] = <function loads>) \rightarrow WebAppInitData
```

Parse WebApp init data and return it as WebAppInitData object

This method doesn't make any security check, so you shall not trust to this data, use safe_parse_webapp_init_data instead.

Parameters

- init_data data from frontend to be parsed
- loads

Returns

Validate raw WebApp init data and return it as WebAppInitData object

Raise ValueError when data is invalid

Parameters

- token bot token
- init_data data from frontend to be parsed and validated
- loads

Returns

Types

class aiogram.utils.web_app.WebAppInitData(**extra_data: Any)

This object contains data that is transferred to the Web App when it is opened. It is empty if the Web App was launched from a keyboard button.

Source: https://core.telegram.org/bots/webapps#webappinitdata

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.

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.

2.5.5 Callback answer

Helper for callback query handlers, can be useful in bots with a lot of callback handlers to automatically take answer to all requests.

Simple usage

For use, it is enough to register the inner middleware aiogram.utils.callback_answer. CallbackAnswerMiddleware in dispatcher or specific router:

```
dispatcher.callback_query.middleware(CallbackAnswerMiddleware())
```

After that all handled callback queries will be answered automatically after processing the handler.

Advanced usage

In some cases you need to have some non-standard response parameters, this can be done in several ways:

Global defaults

Change default parameters while initializing middleware, for example change answer to pre mode and text "OK":

```
dispatcher.callback_query.middleware(CallbackAnswerMiddleware(pre=True, text="OK"))
```

Look at aiogram.utils.callback_answer.CallbackAnswerMiddleware to get all available parameters

Handler specific

By using flags you can change the behavior for specific handler

```
@router.callback_query(<filters>)
@flags.callback_answer(text="Thanks", cache_time=30)
async def my_handler(query: CallbackQuery):
...
```

Flag arguments is the same as in aiogram.utils.callback_answer.CallbackAnswerMiddleware with additional one disabled to disable answer.

A special case

It is not always correct to answer the same in every case, so there is an option to change the answer inside the handler. You can get an instance of <code>aiogram.utils.callback_answer.CallbackAnswer</code> object inside handler and change whatever you want.

\text{\text{Danger}}

Note that is impossible to change callback answer attributes when you use pre=True mode.

Combine that all at once

For example you want to answer in most of cases before handler with text "" but at some cases need to answer after the handler with custom text, so you can do it:

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, 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)}</b>!",
    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
- body
- marker

Returns

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

Wrap elements as section with numbered list

Parameters

- title
- · body
- start
- fmt

Returns

```
aiogram.utils.formatting.as_key_value(key: Any, value: Any) \rightarrow Text
Wrap elements pair as key-value line. (<b>{key}:</b> {value})
```

Parameters

- key
- value

Returns

Text

and lets complete them all:

```
content = as_list(
    as_marked_section(
        Bold("Success:"),
        "Test 1",
        "Test 3",
        "Test 4",
        marker=" ",
    ),
    as_marked_section(
        Bold("Failed:"),
        "Test 2",
        marker=" ",
    ),
    as_marked_section(
        Bold("Summary:"),
        as_key_value("Total", 4),
        as_key_value("Success", 3),
        as_key_value("Failed", 1),
        marker=" ",
    HashTag("#test"),
    sep="\n\n",
)
```

Will be rendered into:

Success:

Test 1

Test 3

Test 4

Failed:

Test 2

Summary:

Total: 4

Success: 3

Failed: 1

#test

Or as HTML:

Available methods

Parameters

- text_key
- entities_key
- replace_parse_mode

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

· parse_mode_key

Returns

```
\textbf{as\_html}() \to 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.



A Warning

The value should always start with '\$' symbol

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums. message_entity_type.MessageEntityType.CASHTAG

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

Bases: Text

Bot command element.



Warning

The value should always start with '/' symbol

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums. message_entity_type.MessageEntityType.BOT_COMMAND

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

Bases: Text

Url element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums. message_entity_type.MessageEntityType.URL

```
class aiogram.utils.formatting.Email(*body: Any, **params: Any)
     Bases: Text
     Email element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.EMAIL
class aiogram.utils.formatting.PhoneNumber(*body: Any, **params: Any)
     Bases: Text
     Phone number element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.PHONE_NUMBER
class aiogram.utils.formatting.Bold(*body: Any, **params: Any)
     Bases: Text
     Bold element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.BOLD
class aiogram.utils.formatting.Italic(*body: Any, **params: Any)
     Bases: Text
     Italic element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.ITALIC
class aiogram.utils.formatting.Underline(*body: Any, **params: Any)
     Bases: Text
     Underline element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.UNDERLINE
class aiogram.utils.formatting.Strikethrough(*body: Any, **params: Any)
     Bases: Text
     Strikethrough element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.STRIKETHROUGH
class aiogram.utils.formatting.Spoiler(*body: Any, **params: Any)
     Bases: Text
     Spoiler element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.SPOILER
class aiogram.utils.formatting.Code(*body: Any, **params: Any)
     Bases: Text
     Code element.
     Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.
     message_entity_type.MessageEntityType.CODE
```

class aiogram.utils.formatting.**Pre**(*body: Any, language: str | None = None, **params: Any)

Bases: Text

Pre element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.message_entity_type.MessageEntityType.PRE

class aiogram.utils.formatting.TextLink(*body: Any, url: str, **params: Any)

Bases: Text

Text link element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.message_entity_type.MessageEntityType.TEXT_LINK

class aiogram.utils.formatting.TextMention(*body: Any, user: User, **params: Any)

Bases: Text

Text mention element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.message_entity_type.MessageEntityType.TEXT_MENTION

class aiogram.utils.formatting.**CustomEmoji**(*body: Any, custom_emoji_id: str, **params: Any)

Bases: Text

Custom emoji element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.message_entity_type.MessageEntityType.CUSTOM_EMOJI

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

Bases: Text

Block quote element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.message_entity_type.MessageEntityType.BLOCKQUOTE

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

Bases: Text

Expandable block quote element.

Will be wrapped into aiogram.types.message_entity.MessageEntity with type aiogram.enums.message_entity_type.MessageEntityType.EXPANDABLE_BLOCKQUOTE

2.5.7 Media group builder

This module provides a builder for media groups, it can be used to build media groups for aiogram.types.input_media_photo.InputMediaPhoto, aiogram.types.input_media_video.InputMediaVideo, aiogram.types.input_media_document.InputMediaDocument and aiogram.types.input_media_audio.InputMediaAudio.

Warning

 $aiogram.types.input_media_animation.InputMediaAnimation$ is not supported yet in the Bot API to send as media group.

Usage

```
media_group = MediaGroupBuilder(caption="Media group caption")

# Add photo
media_group.add_photo(media="https://picsum.photos/200/300")

# Dynamically add photo with known type without using separate method
media_group.add(type="photo", media="https://picsum.photos/200/300")

# ... or video
media_group.add(type="video", media=FSInputFile("media/video.mp4"))
```

To send media group use aiogram.methods.send_media_group.SendMediaGroup() method, but when you use aiogram.utils.media_group.MediaGroupBuilder you should pass media argument as media_group.build().

If you specify caption in aiogram.utils.media_group.MediaGroupBuilder it will be used as caption for first media in group.

```
await bot.send_media_group(chat_id=chat_id, media=media_group.build())
```

References

- add(*, type: Literal[InputMediaType.AUDIO], media: str | InputFile, caption: str | None = None, parse_mode: str | None = UNSET_PARSE_MODE, caption_entities: List[MessageEntity] | None = None, duration: int | None = None, performer: str | None = None, title: str | None = None, **kwargs: Any) → None
- **add**(*, type: Literal[InputMediaType.PHOTO], media: str | InputFile, caption: str | None = None, parse_mode: str | None = UNSET_PARSE_MODE, caption_entities: List[MessageEntity] | None = None, has_spoiler: bool | None = None, **kwargs: Any) → None
- add(*, type: Literal[InputMediaType.VIDEO], media: str | InputFile, thumbnail: InputFile | str | None = None, caption: str | None = None, parse_mode: str | None = UNSET_PARSE_MODE, caption_entities: List[MessageEntity] | None = None, width: int | None = None, height: int | None = None, duration: int | None = None, supports_streaming: bool | None = None, has_spoiler: bool | None = None, **kwargs: Any) → None
- $\begin{array}{l} \textbf{add}(*, type: Literal[InputMediaType.DOCUMENT], media: str \mid InputFile, thumbnail: InputFile \mid str \mid None \\ = None, caption: str \mid None = None, parse_mode: str \mid None = UNSET_PARSE_MODE, caption_entities: \\ List[MessageEntity] \mid None = None, disable_content_type_detection: bool \mid None = None, **kwargs: \\ Any) \rightarrow \text{None} \end{array}$

Add a media object to the media group.

Parameters

kwargs – Keyword arguments for the media object. The available keyword arguments depend on the media type.

Returns

None

add_audio(media: str | ~aiogram.types.input_file.InputFile, thumbnail: ~aiogram.types.input_file.InputFile | None = None, caption: str | None = None, parse_mode: str | None = <Default('parse_mode')>, caption_entities: ~typing.List[~aiogram.types.message_entity.MessageEntity] | None = None, duration: int | None = None, performer: str | None = None, title: str | None = None, **kwargs: ~typing.Any) → None

Add an audio file to the media group.

Parameters

• **media** – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file_attach_name>' to upload a new one using multipart/form-data under <file_attach_name> name.

More information on Sending Files »

- **thumbnail** *Optional*. Thumbnail of the file sent; can be ignored if thumbnail generation for the file is supported server-side. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 320.
- caption Optional. Caption of the audio to be sent, 0-1024 characters after entities parsing
- parse_mode Optional. Mode for parsing entities in the audio caption. See formatting options for more details.
- **caption_entities** *Optional*. List of special entities that appear in the caption, which can be specified instead of *parse_mode*
- duration Optional. Duration of the audio in seconds
- **performer** *Optional*. Performer of the audio
- title Optional. Title of the audio

Returns

None

add_document(media: str | ~aiogram.types.input_file.InputFile, thumbnail:

~aiogram.types.input_file.InputFile | None = None, caption: str | None = None, parse_mode: str | None = <Default('parse_mode')>, caption_entities: ~typing.List[~aiogram.types.message_entity.MessageEntity] | None = None, disable_content_type_detection: bool | 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 & alogram.types.input\_file.InputFile, caption: str} \mid None = None, parse\_mode: str} \mid None = & \langle Default('parse\_mode') \rangle, caption\_entities: \\ & \sim & typing.List[\sim & alogram.types.message\_entity.MessageEntity] \mid None = None, has\_spoiler: bool \mid None = None, **kwargs: ~typing.Any) \rightarrow None \\ \end{tabular}
```

Add a photo to the media group.

Parameters

• **media** – File to send. Pass a file_id to send a file that exists on the Telegram servers (recommended), pass an HTTP URL for Telegram to get a file from the Internet, or pass 'attach://<file_attach_name>' to upload a new one using multipart/form-data under <file_attach_name> name.

More information on Sending Files »

- **caption** *Optional*. Caption of the photo to be sent, 0-1024 characters after entities parsing
- parse_mode *Optional*. Mode for parsing entities in the photo caption. See formatting options for more details.
- **caption_entities** *Optional*. List of special entities that appear in the caption, which can be specified instead of *parse_mode*
- has_spoiler Optional. Pass True if the photo needs to be covered with a spoiler animation

Returns

None

 $\begin{tabular}{ll} \textbf{add_video} (\textit{media: str} \mid \sim & aiogram.types.input_file.InputFile, thumbnail: \sim & aiogram.types.input_file.InputFile \mid \\ None = None, caption: str \mid None = None, parse_mode: str \mid None = < Default('parse_mode') >, \\ & caption_entities: \sim & typing.List[\sim & aiogram.types.message_entity.MessageEntity] \mid None = None, \\ & width: int \mid None = None, height: int \mid None = None, duration: int \mid None = None, \\ & supports_streaming: bool \mid None = None, has_spoiler: bool \mid None = None, **kwargs: \\ & \sim & typing.Any) \rightarrow None \\ \end{tabular}$

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

async aiogram.utils.deep_linking.create_startgroup_link(bot: Bot, payload: str, encode: bool = False, encoder: Callable[[bytes], bytes] | None = None) \rightarrow str

Create 'startgroup' 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

async aiogram.utils.deep_linking.create_startapp_link(bot: Bot, payload: str, encode: bool = False, app_name: str | None = None, encoder: $Callable[[bytes], bytes] \mid None = None) \rightarrow str$

Create 'startapp' deep link with your payload.

If you need to encode payload or pass special characters - set encode as True

Read more:

- Main Mini App links
- Direct mini app links

Parameters

- **bot** bot instance
- payload args passed with /start
- **encode** encode payload with base64url or custom encoder
- app_name if you want direct mini app link
- **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.

2.5.9 Telegram object serialization

Serialization

To serialize Python object to Telegram object you can use pydantic serialization methods, for example:

```
message_data = { ... } # Some message data as dict
message = Message.model_validate(message_data)
```

If you want to bind serialized object to the Bot instance, you can use context:

```
message_data = { ... } # Some message data as dict
message = Message.model_validate(message_data, context={"bot": bot})
```

Deserialization

In cases when you need to deserialize Telegram object to Python object, you can use this module.

To convert Telegram object to Python object excluding files you can use aiogram.utils.serialization. deserialize_telegram_object_to_python() function.

aiogram.utils.serialization.deserialize_telegram_object_to_python(obj: Any, default:

```
DefaultBotProperties | None = None,
include_api_method_name:
bool = True) \rightarrow Any
```

Deserialize telegram object to JSON compatible Python object excluding files.

Parameters

- **obj** The telegram object to be deserialized.
- **default** Default bot properties should be passed only if you want to use custom defaults.
- include_api_method_name Whether to include the API method name in the result.

Returns

The deserialized telegram object.

To convert Telegram object to Python object including files you can use aiogram.utils.serialization. deserialize_telegram_object() function, which returns aiogram.utils.serialization. DeserializedTelegramObject object.

```
aiogram.utils.serialization.deserialize_telegram_object(obj: Any, default: DefaultBotProperties | None = None, include_api_method_name: bool = True) \rightarrow DeserializedTelegramObject
```

Deserialize Telegram Object to JSON compatible Python object.

Parameters

- **obj** The object to be deserialized.
- **default** Default bot properties should be passed only if you want to use custom defaults.
- include_api_method_name Whether to include the API method name in the result.

Returns

The deserialized Telegram object.

class aiogram.utils.serialization.DeserializedTelegramObject(data: Any, files: Dict[str, InputFile])
 Represents a dumped Telegram object.

Parameters

- data (Any) The dumped data of the Telegram object.
- **files** (*Dict[str*, InputFile]) The dictionary containing the file names as keys and the corresponding *InputFile* objects as values.

2.6 Changelog

2.6.1 3.19.0 (2025-03-19)

Features

- Added TypedDict definitions for middleware context data to the dispatcher dependency injection docs.
 - So, now you can use aiogram.dispatcher.middleware.data.MiddlewareData directly or extend it with your own data in the middlewares. #1637
- Added new method aiogram.utils.deep_linking.create_startapp_link() to deep-linking module for creating "startapp" deep links. See also https://core.telegram.org/api/links#main-mini-app-links and https:// core.telegram.org/api/links#direct-mini-app-links #1648, #1651

Bugfixes

- Fixed handling of default empty string ("") in CallbackData filter #1493
- Resolved incorrect ordering of registered handlers in the aiogram.fsm.scene.Scene object caused by
 inspect.getmembers returning sorted members. Handlers are now registered in the order of their definition
 within the class, ensuring proper execution sequence, especially when handling filters with different levels of
 specificity.

For backward compatibility, the old behavior can be restored by setting the attrs_resolver=inspect_members_resolver parameter in the aiogram.fsm.scene.Scene:

```
from aiogram.utils.class_attrs_resolver import inspect_members_resolver

class MyScene(Scene, attrs_resolver=inspect_members_resolver):
```

In this case, the handlers will be registered in the order returned by inspect.getmembers.

By default, the attrs_resolver parameter is set to get_sorted_mro_attrs_resolver now, so you don't need to specify it explicitly. #1641

Improved Documentation

• Updated Ukrainian docs translation #1650

Misc

• Introduce Union types for streamlined type handling.

Implemented Union types across various modules to consolidate and simplify type annotations. This change replaces repetitive union declarations with reusable Union aliases, improving code readability and maintainability. #1592

2.6.2 3.18.0 (2025-02-16)

Features

- Added full support for the Bot API 8.3:
 - Added the parameter chat_id to the method aiogram.methods.send_gift.SendGift, allowing bots
 to send gifts to channel chats.
 - Added the field can_send_gift to the class aiogram.types.chat_full_info.ChatFullInfo.
 - Added the class aiogram.types.transaction_partner_chat.TransactionPartnerChat describing transactions with chats.
 - Added the fields cover and start_timestamp to the class aiogram.types.video.Video, containing a message-specific cover and a start timestamp for the video.
 - Added the parameters cover and start_timestamp to the method aiogram.methods.send_video.
 SendVideo, allowing bots to specify a cover and a start timestamp for the videos they send.
 - Added the fields cover and start_timestamp to the classes aiogram.types.input_media_video. InputMediaVideo and aiogram.types.input_paid_media_video.InputPaidMediaVideo, allowing bots to edit video covers and start timestamps, and specify them for videos in albums and paid media.
 - Added the parameter video_start_timestamp to the methods aiogram.methods.forward_message.
 ForwardMessage and aiogram.methods.copy_message.CopyMessage, allowing bots to change the start timestamp for forwarded and copied videos.
 - Allowed adding reactions to most types of service messages.

#1638

Bugfixes

- Fixed endless loop while adding buttons to the KeyboardBuilder. #1595
- Change the Downloadable protocol to be non-writable to shut up type checking that checks code that uses the bot.download(...) method #1628
- Fix the regex pattern that finds the "bad characters" for deeplink payload. #1630

2.6. Changelog 667

Improved Documentation

- Update data: Dict[Any, str] to data: Dict[str, Any] #1634
- Fix small typo in the Scenes documentation #1640

Misc

- Removed redundant Path to str convertion on file download. #1612
- Increased max redis version support from "<5.1.0" to "<5.3.0" #1631

2.6.3 3.17.0 (2025-01-02)

Features

- Added full support of the Bot API 8.2
 - Added the methods aiogram.methods.verify_user.VerifyUser, aiogram.methods.verify_chat.VerifyChat, aiogram.methods.remove_user_verification.
 RemoveUserVerification and aiogram.methods.remove_chat_verification.
 RemoveChatVerification, allowing bots to manage verifications on behalf of an organization.
 - Added the field upgrade_star_count to the class aiogram.types.gift.Gift.
 - Added the parameter pay_for_upgrade to the method aiogram.methods.send_gift.SendGift.
 - Removed the field hide_url from the class aiogram.types.inline_query_result_article. InlineQueryResultArticle. Pass an empty string as url instead.

#1623

2.6.4 3.16.0 (2024-12-21)

Features

- Added full support of Bot API 8.1:
 - Added the field nanostar_amount to the class aiogram.types.star_transaction.
 StarTransaction.
 - Added the class aiogram.types.transaction_partner_affiliate_program.
 TransactionPartnerAffiliateProgram for transactions pertaining to incoming affiliate commissions.
 - Added the class aiogram.types.affiliate_info.AffiliateInfo and the field affiliate to the class aiogram.types.transaction_partner_user.TransactionPartnerUser, allowing bots to identify the relevant affiliate in transactions with an affiliate commission.

#1617

Bugfixes

• Corrected the exception text of aiogram.methods.base.TelegramMethod._await_ method. #1616

Misc

- Increased max pydantic version support from "<2.10" to "<2.11" #1607
- Fixed closing tag for tg-emoji in the aiogram.utils.text_decoration.HtmlDecoration: use the same constant as for tag opening #1608
- Increased max aiohttp version support from "<3.11" to "<3.12" #1615

2.6.5 3.15.0 (2024-11-17)

Features

- Added full support for Bot API 8.0
 - Added the parameter subscription_period to the method aiogram.methods. create_invoice_link.CreateInvoiceLink to support the creation of links that are billed periodically.
 - Added the parameter business_connection_id to the method aiogram.methods. create_invoice_link.CreateInvoiceLink to support the creation of invoice links on behalf of business accounts.
 - Added the fields subscription_expiration_date, is_recurring and is_first_recurring to the class aiogram.types.successful_payment.SuccessfulPayment.
 - Added the method aiogram.methods.edit_user_star_subscription.EditUserStarSubscription.
 - Added the field subscription_period to the class aiogram.types.transaction_partner_user.
 TransactionPartnerUser.
 - Added the method aiogram.methods.set_user_emoji_status.SetUserEmojiStatus. The user must allow the bot to manage their emoji status.
 - Added the class aiogram.types.prepared_inline_message.PreparedInlineMessage and the
 method aiogram.methods.save_prepared_inline_message.SavePreparedInlineMessage, allowing bots to suggest users send a specific message from a Mini App via the method aiogram.methods.
 share_message.ShareMessage.
 - Added the classes aiogram.types.gift.Gift and aiogram.types.gifts.Gifts and the method aiogram.methods.get_available_gifts.GetAvailableGifts, allowing bots to get all gifts available for sending.
 - Added the field gift to the class aiogram.types.transaction_partner_user.
 TransactionPartnerUser.

#1606

2.6. Changelog 669

2.6.6 3.14.0 (2024-11-02)

Misc

• Checked compatibility with Python 3.13 (added to the CI/CD processes), so now aiogram is totally compatible

Dropped compatibility with Python 3.8 due to this version being EOL.

Warning

In some cases you will need to have the installed compiler (Rust or C++) to install some of the dependencies to compile packages from source on pip install command.

- If you are using Windows, you will need to have the Visual Studio installed.
- If you are using Linux, you will need to have the build-essential package installed.
- If you are using macOS, you will need to have the Xcode installed.

When developers of this dependencies will release new versions with precompiled wheels for Windows, Linux and macOS, this action will not be necessary anymore until the next version of the Python interpreter.

#1589

• Added business_connection_id to the aiogram.types.message.Message API methods shortcuts.

Integrated the business_connection_id attribute into various message manipulation methods, ensuring consistent data handling. This update eliminates the need to pass the business_connection_id as a parameter, instead directly accessing it from the instance attributes. #1586

Features

- Add function get_value to all built-in storage implementations, FSMContext and SceneWizard #1431
- Enhanced the inheritance of handlers and actions in Scenes. Refactored to eliminate the copying of previously connected handlers and actions from parent scenes. Now, handlers are dynamically rebuilt based on the current class, properly utilizing class inheritance and enabling handler overrides.

That's mean that you can now override handlers and actions in the child scene, instead of copying and duplicating them. #1583

- Added full support of Bot API 7.11
 - Added the class aiogram.types.copy_text_button.CopyTextButton and the field copy_text in the class aiogram.types.inline_keyboard_button.InlineKeyboardButton, allowing bots to send and receive inline buttons that copy arbitrary text.
 - Added the parameter allow_paid_broadcast to the methods aiogram.methods.send_message. SendMessage, aiogram.methods.send_photo.SendPhoto, aiogram.methods.send_video. aiogram.methods.send_animation.SendAnimation. SendVideo. aiogram.methods. send_audio.SendAudio. aiogram.methods.send_document.SendDocument, aiogram. methods.send_paid_media.SendPaidMedia, aiogram.methods.send_sticker.SendSticker, aiogram.methods.send_video_note.SendVideoNote, aiogram.methods.send_voice. SendVoice, aiogram.methods.send_location.SendLocation, aiogram.methods.send_venue. SendVenue, aiogram.methods.send_contact.SendContact, aiogram.methods.send_poll. SendPoll. aiogram.methods.send_dice.SendDice, aiogram.methods.send_invoice.

SendInvoice, aiogram.methods.send_game.SendGame, aiogram.methods.send_media_group. SendMediaGroup and aiogram.methods.copy_message.CopyMessage.

- Added the aiogram.types.transaction_partner_telegram_api. TransactionPartnerTelegramApi for transactions related to paid broadcasted messages.
- Introduced the ability to add media to existing text messages using the method aiogram.methods. edit_message_media.EditMessageMedia.
- Added support for hashtag and cashtag entities with a specified chat username that opens a search for the relevant tag within the specified chat.

#1601

Bugfixes

- Fix PytestDeprecationWarning thrown by pytest-asyncio when running the tests #1584
- Fixed customized serialization in the aiogram. filters.callback_data.CallbackData factory.

From now UUID will have 32 bytes length instead of 36 bytes (with no - separators) in the callback data representation. #1602

Improved Documentation

• Add missing closing tag for bold. #1599

2.6.7 3.13.1 (2024-09-18)



Warning

Python 3.8 End of Life: Python 3.8 will reach its end of life (EOL) soon and will no longer be supported by aiogram in the next releases (1-2 months ETA).

Please upgrade to a newer version of Python to ensure compatibility and receive future updates.

Misc

- Increase max pydantic version support "<2.9" -> "<2.10" (only For Python >=3.9) #1576
- Bump aiofiles version upper bound to <24.2 #1577

Bugfixes

• Fixed *Default* object annotation resolution using *pydantic* #1579

2.6. Changelog 671

2.6.8 3.13.0 (2024-09-08)

Features

- Added updates about purchased paid media, represented by the class aiogram.types. paid_media_purchased.PaidMediaPurchased and the field purchased_paid_media in the class aiogram.types.update.Update.
 - Added the ability to specify a payload in aiogram.methods.send_paid_media.SendPaidMedia
 that is received back by the bot in aiogram.types.transaction_partner_user.
 TransactionPartnerUser and purchased_paid_media updates.
 - Added the field prize_star_count to the classes aiogram.types.giveaway_created. GiveawayCreated, aiogram.types.giveaway.Giveaway, aiogram.types. giveaway_winners.GiveawayWinners and aiogram.types.chat_boost_source_giveaway. ChatBoostSourceGiveaway.
 - Added the field is_star_giveaway to the class aiogram.types.giveaway_completed.
 GiveawayCompleted.

#1510

Added missing method aliases such as .answer(), .reply(), and others to InaccessibleMessage. This change ensures consistency and improves usability by aligning the functionality of InaccessibleMessage with the Message type. #1574

Bugfixes

• Fixed link preview options to use global defaults in various types and methods to use global defaults for link_preview_options. This change ensures consistency and enhances flexibility in handling link preview options across different components. #1543

2.6.9 3.12.0 (2024-08-16)

Features

- Added message_thread_id parameter to message.get_url(). #1451
- Added getting user from chat_boost with source ChatBoostSourcePremium in UserContextMiddleware for EventContext #1474
- Added full support of Bot API 7.8
 - Added the ability to send paid media to any chat.
 - Added the parameter business_connection_id to the method aiogram.methods. send_paid_media.SendPaidMedia, allowing bots to send paid media on behalf of a business account.
 - Added the field paid_media to the class aiogram.types.transaction_partner_user.
 TransactionPartnerUser for transactions involving paid media.
 - Added the method aiogram.methods.create_chat_subscription_invite_link.
 CreateChatSubscriptionInviteLink, allowing bots to create subscription invite links.
 - Added the method aiogram.methods.edit_chat_subscription_invite_link.
 EditChatSubscriptionInviteLink, allowing bots to edit the name of subscription invite links.

- Added the field until_date to the class aiogram.types.chat_member_member.ChatMemberMember for members with an active subscription.
- Added support for paid reactions and the class aiogram.types.reaction_type_paid.
 ReactionTypePaid.

#1560

Misc

• Improved performance of StatesGroup #1507

2.6.10 3.11.0 (2024-08-09)

Features

- Added full support of Bot API 7.8
 - Added the field has_main_web_app to the class aiogram.types.user.User, which is returned in the response to aiogram.methods.get_me.GetMe.
 - Added the parameter business_connection_id to the methods aiogram.methods.
 pin_chat_message.PinChatMessage and aiogram.methods.unpin_chat_message.
 UnpinChatMessage, allowing bots to manage pinned messages on behalf of a business account.

#1551

Bugfixes

• Fixed URL path in the "Open" button at the "demo/sendMessage" endpoint in the web_app example. #1546

Misc

- Added method aiogram.types.message.Message.as_reply_parameters(). Replaced usage of the argument reply_to_message_id with reply_parameters in all Message reply methods. #1538
- Added aiohttp v3.10 `support. #1548

2.6.11 3.10.0 (2024-07-07)

Features

- Added full support of Bot API 7.7
 - Added the class aiogram.types.refunded_payment.RefundedPayment, containing information about a refunded payment.
 - Added the field refunded_payment to the class aiogram.types.message.Message, describing a service message about a refunded payment.

#1536

2.6. Changelog 673

2.6.12 3.9.0 (2024-07-06)

Features

- Added ChatMember resolution tool and updated 2.x migration guide. #1525
- Added full support of Bot API 7.6
 - Added the classes aiogram.types.paid_media.PaidMedia,

aiogram.types.paid_media_info.PaidMediaInfo, aiogram.types.paid_media_preview. PaidMediaPreview, aiogram.types.paid_media_photo.PaidMediaPhoto and aiogram.types.paid_media_video.PaidMediaVideo, containing information about paid media.

- Added the method aiogram.methods.send_paid_media.SendPaidMedia

and the classes aiogram.types.input_paid_media.InputPaidMedia, aiogram.types.input_paid_media_photo.InputPaidMediaPhoto and aiogram.types.input_paid_media_video.InputPaidMediaVideo, to support sending paid media.

Documented that the methods aiogram.methods.copy_message.CopyMessage

and aiogram.methods.copy_messages.CopyMessages cannot be used to copy paid media.

- Added the field can_send_paid_media to the class

aiogram.types.chat_full_info.ChatFullInfo.

Added the field paid_media to the classes

aiogram.types.message.Message and aiogram.types.external_reply_info.

- Added the class

aiogram.types.transaction_partner_telegram_ads.TransactionPartnerTelegramAds, containing information about Telegram Star transactions involving the Telegram Ads Platform.

Added the field invoice_payload to the class

aiogram.types.transaction_partner_user.TransactionPartnerUser, containing the bot-specified invoice payload.

- Changed the default opening mode for Direct Link Mini Apps.
- Added support for launching Web Apps via t.me link in the class

aiogram.types.menu_button_web_app.MenuButtonWebApp.

Added the field section_separator_color to the class ThemeParams.

#1533

Bugfixes

Fixed event context resolving for the callback query that is coming from the business account #1520

2.6.13 3.8.0 (2024-06-19)

Features

- Added utility to safely deserialize any Telegram object or method to a JSON-compatible object (dict). (>> Read more) #1450
- Added full support of Bot API 7.5
 - Added the classes aiogram.types.star_transactions.StarTransactions,

```
aiogram.types.star_transaction.StarTransaction,aiogram.types.transaction_partner.TransactionPartnerandaiogram.types.revenue_withdrawal_state.RevenueWithdrawalState,containinginformationaboutTelegram Star transactions involving the bot.
```

- Added the method aiogram.methods.get_star_transactions.GetStarTransactions
 that can be used to get the list of all Telegram Star transactions for the bot.
- Added support for callback buttons in

aiogram.types.inline_keyboard_markup.InlineKeyboardMarkup for messages sent on behalf of a business account.

 Added support for callback queries originating from a message sent on behalf of a business account.

- Added the parameter business_connection_id to the methods

```
aiogram.methods.edit_message_text.EditMessageText, aiogram.methods.edit_message_media.EditMessageMedia, aiogram.methods.edit_message_caption.

EditMessageCaption, aiogram.methods.edit_message_live_location.

EditMessageLiveLocation, aiogram.methods.stop_message_live_location.

StopMessageLiveLocation and aiogram.methods.edit_message_reply_markup.

EditMessageReplyMarkup, allowing the bot to edit business messages.
```

- Added the parameter business_connection_id to the method

aiogram.methods.stop_poll.StopPoll, allowing the bot to stop polls it sent on behalf of a business account.

#1518

Bugfixes

Increased DNS cache ttl setting to aiohttp session as a workaround for DNS resolution issues in aiohttp. #1500

Improved Documentation

- Fixed MongoStorage section in the documentation by adding extra dependency to ReadTheDocs configuration.
 #1501
- Added information about dependency changes to the 2.x --> 3.x migration guide. #1504

2.6. Changelog 675

Misc

- [Only for contributors] Fail redis and mongo tests if incorrect URI provided + some storages tests refactoring
 If incorrect URIs provided to "-redis" and/or "-mongo" options tests should fail with errors instead of skipping.
 Otherwise the next scenario is possible:
 - 1) developer breaks RedisStorage and/or MongoStorage code
 - 2) tests are run with incorrect redis and/or mongo URIsprovided by "-redis" and "-mongo" options (for example, wrong port specified)
 - 3) tests pass because skipping doesn't fail tests run
 - 4) developer or reviewer doesn't notice that redis and/or mongo tests were skipped
 - 5) broken code gets in codebase

Also some refactorings done (related with storages and storages tests). #1510

2.6.14 3.7.0 (2024-05-31)

Features

- Added new storage aiogram.fsm.storage.MongoStorage for Finite State Machine based on Mongo DB (using motor library) #1434
- Added full support of Bot API 7.4 #1498

Bugfixes

• Fixed wrong MarkdownV2 custom emoji parsing in aiogram.utils.text_decorations #1496

Deprecations and Removals

• Removed deprecated arguments from Bot class parse_mode, disable_web_page_preview, protect_content as previously announced in v3.4.0. #1494

Misc

• Improved code consistency and readability in code examples by refactoring imports, adjusting the base webhook URL, modifying bot instance initialization to utilize DefaultBotProperties, and updating router message handlers. #1482

2.6.15 3.6.0 (2024-05-06)

Features

• Added full support of Bot API 7.3 #1480

Improved Documentation

• Added telegram objects transformation block in 2.x -> 3.x migration guide #1412

2.6.16 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. Changelog 677

2.6.17 3.4.1 (2024-02-17)

Bugfixes

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

2.6.18 3.4.0 (2024-02-16)

Features

• Reworked bot-wide globals like parse_mode, disable_web_page_preview, and others to be more flexible.



Warning

Note that the old way of setting these global bot properties is now deprecated and will be removed in the next major release.

#1392

- A new enum KeyboardButtonPollTypeType for KeyboardButtonPollTypeType.type field has bed added. #1398
- Added full support of Bot API 7.1
 - Added support for the administrator rights can_post_stories, can_edit_stories, can_delete_stories in supergroups.
 - Added the class ChatBoostAdded and the field boost_added to the class Message for service messages about a user boosting a chat.
 - Added the field sender_boost_count to the class Message.
 - Added the field reply_to_story to the class Message.
 - Added the fields chat and id to the class Story.
 - Added the field unrestrict_boost_count to the class Chat.
 - Added the field custom_emoji_sticker_set_name to the class Chat.

#1417

Bugfixes

- Update KeyboardBuilder utility, fixed type-hints for button method, adjusted limits of the different markup types to real world values. #1399
- Added new reply_parameters param to message.send_copy because it hasn't been added there #1403

Improved Documentation

• Add notion "Working with plural forms" in documentation Utils -> Translation #1395

2.6.19 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

2.6.20 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

2.6.21 3.1.1 (2023-09-25)

Bugfixes

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

2.6.22 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

2.6.23 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

2.6.24 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

2.6.25 3.0.0rc1 (2023-08-06)

Features

• Added Currency enum. You can use it like this:

```
from aiogram.enums import Currency
await bot.send_invoice(
    ...,
    currency=Currency.USD,
    ...
)
```

#1194

- Updated keyboard builders with new methods for integrating buttons and keyboard creation more seamlessly.
 Added functionality to create buttons from existing markup and attach another builder. This improvement aims to make the keyboard building process more user-friendly and flexible. #1236
- Added support for message_thread_id in ChatActionSender #1249

Bugfixes

- Fixed polling startup when "bot" key is passed manually into dispatcher workflow data #1242
- Added codegen configuration for lost shortcuts:
 - ShippingQuery.answer
 - PreCheckoutQuery.answer
 - Message.delete_reply_markup

#1244

Improved Documentation

• Added documentation for webhook and polling modes. #1241

Misc

- Reworked InputFile reading, removed __aiter__ method, added *bot: Bot* argument to the .read(...) method, so, from now URLInputFile can be used without specifying bot instance. #1238
- Code-generated __init__ typehints in types and methods to make IDE happy without additional pydantic plugin #1245

2.6.26 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

2.6.27 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:

(continued from previous page)

```
storage=..., # Any persistent storage
)
```

1 Note

If you have implemented you own storages you should extend record key generation with new one attribute - thread_id

#1161

- Improved CallbackData serialization.
 - Minimized UUID (hex without dashes)
 - Replaced bool values with int (true=1, false=0)

#1163

- Added a tool to make text formatting flexible and easy. More details on the corresponding documentation page #1172
- Added X-Telegram-Bot-Api-Secret-Token header check #1173
- Made allowed_updates list to revolve automatically in start polling method if not set explicitly. #1178
- Added possibility to pass custom headers to URLInputFile object #1191

Bugfixes

• Change type of result in InlineQueryResult enum for InlineQueryResultCachedMpeg4Gif and InlineQueryResultMpeg4Gif to more correct according to documentation.

Change regexp for entities parsing to more correct (InlineQueryResultType.yml). #1146

- Fixed signature of startup/shutdown events to include the **dispatcher.workflow_data as the handler arguments. #1155
- Added missing FORUM_TOPIC_EDITED value to content_type property #1160
- Fixed compatibility with Python 3.8-3.9 (from previous release) #1162
- Fixed the markdown spoiler parser. #1176
- Fixed workflow data propagation #1196
- Fixed the serialization error associated with nested subtypes like InputMedia, ChatMember, etc.

The previously generated code resulted in an invalid schema under pydantic v2, which has stricter type parsing. Hence, subtypes without the specification of all subtype unions were generating an empty object. This has been rectified now. #1213

Improved Documentation

Changed small grammar typos for upload_file #1133

Deprecations and Removals

• Removed text filter in due to is planned to remove this filter few versions ago.

Use F. text instead #1170

Misc

• Added full support of Bot API 6.6



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



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



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

2.6.28 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())
```

Features

• Added missing shortcuts, new enums, reworked old stuff

Breaking All previously added enums is re-generated in new place - aiogram.enums instead of aiogram.types

Added enums: aiogram.enums.bot_command_scope_type.BotCommandScopeType,

```
aiogram.enums.chat_action.ChatAction,
                                               aiogram.enums.chat_member_status.
ChatMemberStatus,
                         aiogram.enums.chat_type.ChatType,
                                                                  aiogram.enums.
content_type.ContentType,
                            aiogram.enums.dice_emoji.DiceEmoji,
                                                                  aiogram.enums.
inline_query_result_type.InlineQueryResultType,
                                                 aiogram.enums.input_media_type.
                 aiogram.enums.mask_position_point.MaskPositionPoint,
InputMediaType,
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. aiogram.types.chat.Chat.create_invite_link(), export_invite_link(), aiogram.types.chat.Chat.do(), aiogram.types.chat.Chat. delete_sticker_set(), aiogram.types.chat.Chat.set_sticker_set(), aiogram.types. chat.Chat.get_member(), aiogram.types.chat.Chat.get_member_count(), aiogram. types.chat.Chat.leave(), aiogram.types.chat.Chat.unpin_all_messages(), aiogram. types.chat.Chat.unpin_message(), aiogram.types.chat.Chat.pin_message(), aiogram.types.chat.Set_administrator_custom_title(), aiogram.types.chat. Chat.set_permissions(), aiogram.types.chat.Chat.promote(), aiogram.types.chat. Chat.restrict(), aiogram.types.chat.Chat.unban(), aiogram.types.chat.Chat. aiogram.types.chat.Chat.set_description(), aiogram.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 <code>aiogram.types.chat_permissions.ChatPermissions</code> is updated without backward compatibility, so now this object has no <code>can_send_media_messages</code> 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

2.6.29 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

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

2.6.30 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

2.6.31 3.0.0b4 (2022-08-14)

Features

- Add class helper ChatAction for constants that Telegram BotAPI uses in sendChatAction request. In my opinion, this will help users and will also improve compatibility with 2.x version where similar class was called "ChatActions". #803
- · Added possibility to combine filters or invert result

Example:

```
Text(text="demo") | Command(commands=["demo"])
MyFilter() & AnotherFilter()
~StateFilter(state='my-state')
```

#894

- Fixed type hints for redis TTL params. #922
- Added full_name shortcut for Chat object #929

Bugfixes

- Fixed false-positive coercing of Union types in API methods #901
- Added 3 missing content types:
 - proximity_alert_triggered
 - supergroup_chat_created
 - channel chat created

#906

- Fixed the ability to compare the state, now comparison to copy of the state will return True. #927
- Fixed default lock kwargs in RedisEventIsolation. #972

Misc

- Restrict including routers with strings #896
- Changed CommandPatterType to CommandPatternType in aiogram/dispatcher/filters/command.py #907
- Added full support of Bot API 6.1 #936
- Breaking! More flat project structure

These packages was moved, imports in your code should be fixed:

- aiogram.dispatcher.filters -> aiogram.filters
- aiogram.dispatcher.fsm -> aiogram.fsm
- aiogram.dispatcher.handler -> aiogram.handler
- aiogram.dispatcher.webhook -> aiogram.webhook
- aiogram.dispatcher.flags/* -> aiogram.dispatcher.flags (single module instead of package)

#938

- Removed deprecated router.<event>_handler and router.register_<event>_handler methods. #941
- Deprecated filters factory. It will be removed in next Beta (3.0b5) #942
- MessageEntity method get_text was removed and extract was renamed to extract_from #944
- Added full support of Bot API 6.2 #975

2.6.32 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

2.6.33 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

2.6.34 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

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

2.6.35 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

2.6.36 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

2.6.37 3.0.0a16 (2021-09-22)

Features

- Added support of local Bot API server files downloading
 When Local API is enabled files can be downloaded via bot.download/bot.download_file methods. #698
- Implemented I18n & L10n support #701

Misc

- Covered by tests and docs KeyboardBuilder util #699
- Breaking!!!. Refactored and renamed exceptions.
 - Exceptions module was moved from aiogram.utils.exceptions to aiogram.exceptions
 - Added prefix *Telegram* for all error classes

#700

- Replaced all pragma: no cover marks via global .coveragerc config #702
- Updated dependencies.

Breaking for framework developers Now all optional dependencies should be installed as extra: *poetry install -E fast -E redis -E proxy -E i18n -E docs* #703

2.6.38 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

2.6.39 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

2.6.40 2.14.3 (2021-07-21)

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

2.6.41 2.14.2 (2021-07-26)

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

2.6.42 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)

2.6.43 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)

2.6.44 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)

2.6.45 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)

2.6.46 2.11.2 (2021-11-10)

- · Fixed default parse mode
- Added missing "supports_streaming" argument to answer_video method #462

2.6.47 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)

2.6.48 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)

2.6.49 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'

2.6.50 2.10 (2021-09-13)

- Breaking change: Stop using _MainThread event loop in bot/dispatcher instances (#397)
- Breaking change: Replaced aiomongo with motor (#368, #380)
- Fixed: TelegramObject's aren't destroyed after update handling #307 (#371)
- Add setting current context of Telegram types (#369)
- Fixed markdown escaping issues (#363)
- Fixed HTML characters escaping (#409)
- Fixed italic and underline decorations when parse entities to Markdown
- Fixed #413: parse entities positioning (#414)
- Added missing thumb parameter (#362)
- Added public methods to register filters and middlewares (#370)
- Added ChatType builtin filter (#356)
- Fixed IDFilter checking message from channel (#376)
- Added missed answer_poll and reply_poll (#384)
- Added possibility to ignore message caption in commands filter (#383)
- · Fixed addStickerToSet method
- Added preparing thumb in send_document method (#391)
- Added exception MessageToPinNotFound (#404)
- Fixed handlers parameter-spec solving (#408)
- Fixed CallbackQuery.answer() returns nothing (#420)
- CHOSEN INLINE RESULT is a correct API-term (#415)
- Fixed missing attributes for Animation class (#422)
- Added missed emoji argument to reply_dice (#395)
- Added is_chat_creator method to ChatMemberStatus (#394)

- Added missed ChatPermissions to __all__ (#393)
- Added is_forward method to Message (#390)
- Fixed usage of deprecated is_private function (#421)

and many others documentation and examples changes:

- Updated docstring of RedisStorage2 (#423)
- Updated I18n example (added docs and fixed typos) (#419)
- A little documentation revision (#381)
- Added comments about correct errors_handlers usage (#398)
- Fixed typo rexex -> regex (#386)
- Fixed docs Quick start page code blocks (#417)
- fixed type hints of callback_data (#400)
- Prettify readme, update downloads stats badge (#406)

2.6.51 2.9.2 (2021-06-13)

- Fixed Message.get_full_command() #352
- Fixed markdown util #353

2.6.52 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)

2.6.53 2.8 (2021-04-26)

- Added full support of Bot API 4.8
- Added Message.answer_dice and Message.reply_dice methods (#306)

2.6.54 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

2.6.55 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)

2.6.56 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

2.6.57 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

2.6.58 2.5.2 (2021-01-01)

• Get back quote_html and escape_md functions

2.6.59 2.5.1 (2021-01-01)

• Hot-fix of CommandStart filter

2.6.60 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...)

2.6.61 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

2.6.62 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

2.6.63 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)

2.6.64 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

2.6.65 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)

2.6.66 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

2.6.67 1.4 (2021-08-03)

• Bot API 4.0 (#57)

2.6.68 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.

2.6.69 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

2.6.70 1.3.1 (2018-05-27)

2.6.71 1.3 (2021-04-22)

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

2.6.72 1.2.3 (2018-04-14)

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

2.6.73 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.

2.6.74 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.

2.6.75 1.2 (2018-02-23)

- Full provide Telegram Bot API 3.6
- Fixed critical error: Fatal Python error: PyImport_GetModuleDict: no module dictionary!
- Implemented connection pool in RethinkDB driver
- · Typo fixes of documentstion
- · Other bug fixes and minor improvements.

2.6.76 1.1 (2018-01-27)

- Added more methods for data types (like message.reply_sticker(...) or file.download(...)
- · Typo fixes of documentstion
- 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)

2.6.77 1.0.4 (2018-01-10)

2.6.78 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.

2.6.79 1.0.2 (2017-11-29)

2.6.80 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.

2.6.81 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.

```
2.6.82 0.4.1 (2017-08-03)
```

2.6.83 0.4 (2017-08-05)

2.6.84 0.3.4 (2017-08-04)

2.6.85 0.3.3 (2017-07-05)

2.6.86 0.3.2 (2017-07-04)

2.6.87 0.3.1 (2017-07-04)

2.6.88 0.2b1 (2017-06-00)

2.6.89 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.

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:

2.7. Contributing 707

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

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, StackOverflow 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.

2.7. Contributing 709

PYTHON MODULE INDEX

aiogram.enums.bot_command_scope_type, 537 aiogram.enums.chat_action, 538 aiogram.enums.chat_boost_source_type, 538 aiogram.enums.chat_member_status, 539 aiogram.enums.chat_type, 539 aiogram.enums.chat_type, 539 aiogram.enums.content_type, 539 aiogram.enums.cortent_type, 544 aiogram.enums.injunt_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.pais_media_type, 549 aiogram.enthods.delete_chat_boto, 396 aiogram.enthods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.enums.pais_media_type, 549 aiogram.enthods.delete_message, 389 aiogram.enthods.delete_message, 389 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.enums.dice_moiodos.create_chat_subscription_oiodos.create_chat_subscripti	a	<pre>aiogram.methods.approve_chat_join_request,</pre>
aiogram.enums.bot_command_scope_type, 537 aiogram.enums.chat_action, 538 aiogram.enums.chat_boost_source_type, 538 aiogram.enums.chat_boost_source_type, 538 aiogram.enums.chat_member_status, 539 aiogram.enums.chat_type, 539 aiogram.enums.content_type, 539 aiogram.enums.content_type, 539 aiogram.enums.currency, 541 aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_media_type, 546 aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 548 aiogram.enums.message_entity_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.paid_media_type, 549 aiogram.enums.paid_media_type, 549 aiogram.enums.paid_media_type, 549 aiogram.enums.paid_media_type, 549 aiogram.enums.passport_element_error_type, 549 aiogra	aiogram.dispatcher.flags.620	381
aiogram.enums.chat_action, 538 aiogram.enums.chat_boost_source_type, 538 aiogram.enums.chat_member_status, 539 aiogram.enums.chat_type, 539 aiogram.enums.content_type, 539 aiogram.enums.currency, 541 aiogram.enums.dice_emoji, 544 aiogram.enums.inline_query_result_type, 545 aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.close_forum_topic, 385 aiogram.methods.close_general_forum_topic, 385 aiogram.methods.copy_message, 387 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link, 523 aiogram.methods.create_invoice_link, 523 aiogram.methods.delete_chat_join_request, 395 aiogram.methods.delete_chat_sticker_set, 397 aiogram.enums.paid_media_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_message, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		aiogram.methods.ban_chat_member,382
aiogram.enums.chat_boost_source_type, 538 aiogram.enums.chat_member_status, 539 aiogram.enums.chat_type, 539 aiogram.enums.content_type, 539 aiogram.enums.corrency, 541 aiogram.enums.dice_emoji, 544 aiogram.enums.encrypted_passport_element, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.close_general_forum_topic, 385 aiogram.methods.copy_message, 387 aiogram.methods.copy_message, 387 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link, 523 aiogram.methods.delete_chat_pioin_request, 395 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_my_commands, 399		aiogram.methods.ban_chat_sender_chat,383
aiogram.enums.chat_member_status, 539 aiogram.enums.chat_type, 539 aiogram.enums.content_type, 539 aiogram.enums.currency, 541 aiogram.enums.dice_emoji, 544 aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.menu_button_type, 548 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.close_general_forum_topic, 385 aiogram.methods.copy_message, 387 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link, 523 aiogram.methods.delete_chat_join_request, 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_messages, 498 aiogram.methods.close_general_forum_topic, 385 aiogram.methods.create_forum_topic, 385 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.delete_chat_join_request, 395 aiogram.methods.delete_chat_sticker_set, 358 aiogram.methods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.methods.delete		aiogram.methods.close, 384
aiogram.enums.chat_type, 539 aiogram.enums.content_type, 539 aiogram.enums.currency, 541 aiogram.enums.dice_emoji, 544 aiogram.enums.encrypted_passport_element, 545 aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.mask_position_point, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menubutton_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.message_origin_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.message_origin_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.message_origin_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_message, 498 aiogram.methods.delete_message, 498 aiogram.methods.delete_my_commands, 399		<pre>aiogram.methods.close_forum_topic, 385</pre>
aiogram.enums.content_type, 539 aiogram.enums.currency, 541 aiogram.enums.dice_emoji, 544 aiogram.enums.encrypted_passport_element, 545 aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.menu_button_type, 548 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.passport_element_error_type, 549 386 aiogram.methods.copy_message, 387 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_new_sticker_set, 358 aiogram.methods.delete_chat_join_request, 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		<pre>aiogram.methods.close_general_forum_topic,</pre>
aiogram.enums.currency, 541 aiogram.enums.dice_emoji, 544 aiogram.enums.encrypted_passport_element, 545 aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.paid_media_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.copy_message, 387 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link, 523 aiogram.methods.delete_chat_join_request, 395 aiogram.methods.delete_chat_sticker_set, 397 aiogram.enums.paid_media_type, 549 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_messages, 498 aiogram.methods.create_chat_invite_link, 391 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_chat_subscription_		386
aiogram.enums.dice_emoji,544 aiogram.enums.encrypted_passport_element,545 aiogram.enums.inline_query_result_type,545 aiogram.enums.input_media_type,546 aiogram.enums.input_paid_media_type,546 aiogram.enums.mask_position_point,547 aiogram.enums.mask_position_point,547 aiogram.enums.message_entity_type,548 aiogram.enums.message_origin_type,548 aiogram.enums.paid_media_type,549 aiogram.enums.parse_mode,549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link, 523 aiogram.methods.create_new_sticker_set, 358 aiogram.methods.decline_chat_join_request, 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_my_commands, 399		aiogram.methods.copy_message,387
aiogram.enums.encrypted_passport_element,545 aiogram.enums.inline_query_result_type,545 aiogram.enums.input_media_type,546 aiogram.enums.input_paid_media_type,546 aiogram.enums.mask_position_point,547 aiogram.enums.message_entity_type,548 aiogram.enums.message_entity_type,548 aiogram.enums.message_origin_type,548 aiogram.enums.paid_media_type,549 aiogram.enums.parse_mode,549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link,523 aiogram.methods.create_new_sticker_set,358 aiogram.methods.decline_chat_join_request, 395 aiogram.methods.delete_chat_photo,396 aiogram.methods.delete_chat_sticker_set,397 aiogram.methods.delete_forum_topic,398 aiogram.methods.delete_message,496 aiogram.methods.delete_messages,498 aiogram.methods.delete_messages,498 aiogram.methods.delete_my_commands,399		aiogram.methods.copy_messages,389
aiogram.enums.inline_query_result_type, 545 aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 549 aiogram.methods.create_chat_subscription_invite_link, 392 aiogram.methods.create_forum_topic, 393 aiogram.methods.create_invoice_link, 523 aiogram.methods.create_new_sticker_set, 358 aiogram.methods.decline_chat_join_request, 395 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		<pre>aiogram.methods.create_chat_invite_link, 391</pre>
aiogram.enums.input_media_type, 546 aiogram.enums.input_paid_media_type, 546 aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		<pre>aiogram.methods.create_chat_subscription_invite_link,</pre>
aiogram.enums.input_paid_media_type, 546 aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.create_invoice_link, 523 aiogram.methods.create_new_sticker_set, 358 aiogram.methods.decline_chat_join_request, 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_message, 496 aiogram.methods.delete_message, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		392
aiogram.enums.keyboard_button_poll_type_type, 547 aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.create_invoice_link, 523 aiogram.methods.decline_chat_join_request, 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		aiogram.methods.create_forum_topic,393
aiogram.enums.mask_position_point, 547 aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		
aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		
aiogram.enums.menu_button_type, 547 aiogram.enums.message_entity_type, 548 aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 395 aiogram.methods.delete_chat_photo, 396 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399	aiogram.enums.mask_position_point,547	<pre>aiogram.methods.decline_chat_join_request,</pre>
aiogram.enums.message_origin_type, 548 aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_chat_sticker_set, 397 aiogram.methods.delete_forum_topic, 398 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399	aiogram.enums.menu_button_type, 547	
aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		
aiogram.enums.paid_media_type, 549 aiogram.enums.parse_mode, 549 aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_message, 496 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399		
aiogram.enums.passport_element_error_type, 549 aiogram.methods.delete_messages, 498 aiogram.methods.delete_my_commands, 399	aiogram.enums.paid_media_type, 549	
549 aiogram.methods.delete_my_commands, 399	aiogram.enums.parse_mode, 549	
	<pre>aiogram.enums.passport_element_error_type,</pre>	
	549	
alogiam: Chambi poli_cype, 550	aiogram.enums.poll_type,550	<pre>aiogram.methods.delete_sticker_from_set, 359</pre>
aiogram.enums.reaction_type_type,550 aiogram.methods.delete_sticker_set,360	<pre>aiogram.enums.reaction_type_type, 550</pre>	
aiogram.enums.revenue_withdrawal_state_type, aiogram.methods.delete_webhook, 531	<pre>aiogram.enums.revenue_withdrawal_state_type,</pre>	
aiogram.methods.edit_chat_invite_link, 400	550	
aiogram.enums.sticker_format,551 aiogram.methods.edit_chat_subscription_invite_link,	aiogram.enums.sticker_format,551	
aiogram.enums.sticker_type,551 401	<pre>aiogram.enums.sticker_type, 551</pre>	
aiogram.enums.topic_icon_color,551 aiogram.methods.edit_forum_topic,402	<pre>aiogram.enums.topic_icon_color,551</pre>	
aiogram.enums.transaction_partner_type,552 aiogram.methods.edit_general_forum_topic,404	<pre>aiogram.enums.transaction_partner_type, 552</pre>	
aiogram.enums.update_type, 552 aiogram.methods.edit_message_caption, 499	aiogram.enums.update_type,552	
aiogram.exceptions, 618 aiogram.methods.edit_message_live_location,		-
aiogram.handlers.callback_query,631 500	aiogram.handlers.callback_query,631	
aiogram.methods.add_sticker_to_set, 357 aiogram.methods.edit_message_media, 503	<pre>aiogram.methods.add_sticker_to_set, 357</pre>	
aiogram.methods.answer_callback_query, 379 aiogram.methods.edit_message_reply_markup,	<pre>aiogram.methods.answer_callback_query, 379</pre>	
aiogram.methods.answer_inline_query,510 504	<pre>aiogram.methods.answer_inline_query,510</pre>	
aiogram.methods.answer_pre_checkout_query, aiogram.methods.edit_message_text,505	<pre>aiogram.methods.answer_pre_checkout_query,</pre>	
aiogram.methods.edit_user_star_subscription,		
aiogram.methods.answer_shipping_query, 521 525		·
aiogram.methods.answer_web_app_query, 512 aiogram.methods.export_chat_invite_link, 405	aiogram.methods.answer_web_app_query,512	alogram.methods.export_cnat_invite_link, 405

```
aiogram.methods.send_media_group, 449
aiogram.methods.forward_message, 406
aiogram.methods.forward_messages, 407
                                               aiogram.methods.send_message, 451
aiogram.methods.get_available_gifts, 361
                                               aiogram.methods.send_paid_media, 454
aiogram.methods.get_business_connection, 408
                                               aiogram.methods.send_photo, 456
aiogram.methods.get_chat, 409
                                               aiogram.methods.send_poll, 459
aiogram.methods.get_chat_administrators, 410
                                               aiogram.methods.send_sticker, 367
aiogram.methods.get_chat_member,411
                                               aiogram.methods.send_venue, 463
aiogram.methods.get_chat_member_count, 412
                                               aiogram.methods.send_video, 465
aiogram.methods.get_chat_menu_button, 412
                                               aiogram.methods.send_video_note, 469
aiogram.methods.get_custom_emoji_stickers,
                                               aiogram.methods.send_voice, 471
                                               aiogram.methods.set_chat_administrator_custom_title,
aiogram.methods.get_file, 413
aiogram.methods.get_forum_topic_icon_stickers,aiogram.methods.set_chat_description,475
       414
                                               aiogram.methods.set_chat_menu_button, 476
aiogram.methods.get_game_high_scores,516
                                               aiogram.methods.set_chat_permissions,477
aiogram.methods.get_me, 415
                                               aiogram.methods.set_chat_photo,478
aiogram.methods.get_my_commands, 415
                                               aiogram.methods.set_chat_sticker_set, 479
aiogram.methods.get_my_default_administrator_raiphqpsam.methods.set_chat_title,480
                                               aiogram.methods.set_custom_emoji_sticker_set_thumbnail,
aiogram.methods.get_my_description, 417
aiogram.methods.get_my_name, 418
                                               aiogram.methods.set_game_score, 519
aiogram.methods.get_my_short_description, 419
                                               aiogram.methods.set_message_reaction, 481
aiogram.methods.get_star_transactions, 526
                                               aiogram.methods.set_my_commands, 483
aiogram.methods.get_sticker_set, 363
                                               aiogram.methods.set_my_default_administrator_rights,
{\tt aiogram.methods.get\_updates}, 532
aiogram.methods.get_user_chat_boosts,419
                                               aiogram.methods.set_my_description, 485
aiogram.methods.get_user_profile_photos, 420
                                               aiogram.methods.set_my_name, 486
aiogram.methods.get_webhook_info,533
                                               aiogram.methods.set_my_short_description, 487
aiogram.methods.hide_general_forum_topic, 421
                                               aiogram.methods.set_passport_data_errors,536
                                               aiogram.methods.set_sticker_emoji_list, 371
aiogram.methods.leave_chat, 422
aiogram.methods.log_out, 423
                                               aiogram.methods.set_sticker_keywords, 372
aiogram.methods.pin_chat_message, 424
                                               aiogram.methods.set_sticker_mask_position,
aiogram.methods.promote_chat_member, 425
aiogram.methods.refund_star_payment,527
                                               aiogram.methods.set_sticker_position_in_set,
aiogram.methods.remove_chat_verification, 363
aiogram.methods.remove_user_verification, 364
                                               aiogram.methods.set_sticker_set_thumbnail,
aiogram.methods.reopen_forum_topic, 427
aiogram.methods.reopen_general_forum_topic,
                                               aiogram.methods.set_sticker_set_title, 376
        428
                                               aiogram.methods.set_user_emoji_status,488
aiogram.methods.replace_sticker_in_set, 365
                                               aiogram.methods.set_webhook, 534
aiogram.methods.restrict_chat_member, 429
                                               aiogram.methods.stop_message_live_location,
aiogram.methods.revoke_chat_invite_link, 430
                                                       507
aiogram.methods.save_prepared_inline_message, aiogram.methods.stop_poll, 509
        513
                                               aiogram.methods.unban_chat_member, 489
aiogram.methods.send_animation, 432
                                               aiogram.methods.unban_chat_sender_chat, 490
aiogram.methods.send_audio,435
                                               aiogram.methods.unhide_general_forum_topic,
aiogram.methods.send_chat_action, 437
aiogram.methods.send_contact, 439
                                               aiogram.methods.unpin_all_chat_messages, 492
aiogram.methods.send_dice, 441
                                               aiogram.methods.unpin_all_forum_topic_messages,
aiogram.methods.send_document, 443
aiogram.methods.send_game, 517
                                               aiogram.methods.unpin_all_general_forum_topic_messages,
aiogram.methods.send_gift, 366
aiogram.methods.send_invoice, 528
                                               aiogram.methods.unpin_chat_message, 495
                                               aiogram.methods.upload_sticker_file, 377
aiogram.methods.send_location, 446
```

712 Python Module Index

```
aiogram.methods.verify_chat, 378
                                               aiogram.types.chat_invite_link,57
aiogram.methods.verify_user, 379
                                               aiogram.types.chat_join_request, 58
aiogram.types.affiliate_info, 325
                                               aiogram.types.chat_location, 100
aiogram.types.animation, 19
                                               aiogram.types.chat_member, 100
aiogram.types.audio, 20
                                               aiogram.types.chat_member_administrator, 101
aiogram.types.background_fill, 21
                                               aiogram.types.chat_member_banned, 102
aiogram.types.background_fill_freeform_gradientiogram.types.chat_member_left, 103
                                               aiogram.types.chat_member_member, 103
aiogram.types.background_fill_gradient, 21
                                               aiogram.types.chat_member_owner, 104
aiogram.types.background_fill_solid, 22
                                               aiogram.types.chat_member_restricted, 104
aiogram.types.background_type, 22
                                               aiogram.types.chat_member_updated, 106
                                               aiogram.types.chat_permissions, 127
aiogram.types.background_type_chat_theme, 22
aiogram.types.background_type_fill, 23
                                               aiogram.types.chat_photo, 128
aiogram.types.background_type_pattern, 23
                                               aiogram.types.chat_shared, 129
aiogram.types.background_type_wallpaper, 24
                                               aiogram.types.chosen_inline_result, 278
aiogram.types.birthdate, 24
                                               aiogram.types.contact, 129
aiogram.types.bot_command, 25
                                               aiogram.types.copy_text_button, 130
aiogram.types.bot_command_scope, 25
                                               aiogram.types.dice, 130
aiogram.types.bot_command_scope_all_chat_adminaistgrammrsypes.document, 130
                                               aiogram.types.encrypted_credentials, 344
aiogram.types.bot_command_scope_all_group_chataiogram.types.encrypted_passport_element, 344
                                               aiogram.types.error_event, 617
aiogram.types.bot_command_scope_all_private_chaminsgram.types.external_reply_info, 131
                                               aiogram.types.file, 133
aiogram.types.bot_command_scope_chat, 26
                                               aiogram.types.force_reply, 133
aiogram.types.bot_command_scope_chat_administrantmogram.types.forum_topic, 134
                                               aiogram.types.forum_topic_closed, 134
aiogram.types.bot_command_scope_chat_member,
                                               aiogram.types.forum_topic_created, 135
                                               aiogram.types.forum_topic_edited, 135
aiogram.types.bot_command_scope_default, 28
                                               aiogram.types.forum_topic_reopened, 135
aiogram.types.bot_description, 28
                                               aiogram.types.game, 356
aiogram.types.bot_name, 28
                                               aiogram.types.game_high_score, 357
aiogram.types.bot_short_description, 28
                                               aiogram.types.general_forum_topic_hidden, 135
aiogram.types.business_connection, 29
                                               aiogram.types.general_forum_topic_unhidden,
aiogram.types.business_intro, 29
                                                       136
aiogram.types.business_location, 30
                                               aiogram.types.gift, 340
aiogram.types.business_messages_deleted, 30
                                               aiogram.types.gifts, 340
aiogram.types.business_opening_hours, 30
                                               aiogram.types.giveaway, 136
aiogram.types.business_opening_hours_interval,aiogram.types.giveaway_completed, 137
                                               aiogram.types.giveaway_created, 137
aiogram.types.callback_game, 356
                                               aiogram.types.giveaway_winners, 137
aiogram.types.callback_query, 31
                                               aiogram.types.inaccessible_message, 138
aiogram.types.chat, 32
                                               aiogram.types.inline_keyboard_button, 181
aiogram.types.chat_administrator_rights, 48
                                               aiogram.types.inline_keyboard_markup, 182
aiogram.types.chat_background, 50
                                               aiogram.types.inline_query, 278
                                               aiogram.types.inline_query_result, 279
aiogram.types.chat_boost, 50
aiogram.types.chat_boost_added, 50
                                               aiogram.types.inline_query_result_article,
aiogram.types.chat_boost_removed, 51
                                                       280
                                               aiogram.types.inline_query_result_audio, 282
aiogram.types.chat_boost_source, 51
aiogram.types.chat_boost_source_gift_code, 51
                                               aiogram.types.inline_query_result_cached_audio,
aiogram.types.chat_boost_source_giveaway, 52
aiogram.types.chat_boost_source_premium, 52
                                               aiogram.types.inline_query_result_cached_document,
aiogram.types.chat_boost_updated, 53
                                                       285
aiogram.types.chat_full_info,53
```

Python Module Index 713

```
aiogram.types.inline_query_result_cached_gif,
                                               aiogram.types.keyboard_button_request_user,
aiogram.types.inline_query_result_cached_mpeg4_gif,
                                               aiogram.types.keyboard_button_request_users,
aiogram.types.inline_query_result_cached_photo,
                                                       195
                                               aiogram.types.labeled_price, 326
aiogram.types.inline_query_result_cached_stickariogram.types.link_preview_options, 196
                                               aiogram.types.location, 196
aiogram.types.inline_query_result_cached_videoaiogram.types.login_url, 197
                                               aiogram.types.mask_position, 341
aiogram.types.inline_query_result_cached_voiceaiogram.types.maybe_inaccessible_message, 198
                                               aiogram.types.menu_button, 198
aiogram.types.inline_query_result_contact,
                                               aiogram.types.menu_button_commands, 198
                                               aiogram.types.menu_button_default, 199
aiogram.types.inline_query_result_document,
                                               aiogram.types.menu_button_web_app, 199
                                               aiogram.types.message, 199
aiogram.types.inline_query_result_game, 303
                                               aiogram.types.message_auto_delete_timer_changed,
aiogram.types.inline_query_result_qif, 304
                                                       255
aiogram.types.inline_query_result_location,
                                               aiogram.types.message_entity, 255
                                               aiogram.types.message_id, 256
aiogram.types.inline_query_result_mpeg4_gif,
                                               aiogram.types.message_origin, 256
                                               aiogram.types.message_origin_channel, 257
aiogram.types.inline_query_result_photo, 311
                                               aiogram.types.message_origin_chat, 257
aiogram.types.inline_query_result_venue, 313
                                               aiogram.types.message_origin_hidden_user, 258
aiogram.types.inline_query_result_video, 314
                                               aiogram.types.message_origin_user, 258
aiogram.types.inline_query_result_voice, 316
                                               aiogram.types.message_reaction_count_updated,
aiogram.types.inline_query_results_button,
                                                       259
                                               aiogram.types.message_reaction_updated, 259
aiogram.types.input_contact_message_content,
                                               aiogram.types.order_info, 327
       318
                                               aiogram.types.paid_media, 260
aiogram.types.input_file, 182
                                               aiogram.types.paid_media_info, 260
aiogram.types.input_invoice_message_content,
                                               aiogram.types.paid_media_photo, 261
                                               aiogram.types.paid_media_preview, 261
aiogram.types.input_location_message_content,
                                               aiogram.types.paid_media_purchased, 327
        322
                                               aiogram.types.paid_media_video, 261
aiogram.types.input_media, 183
                                               aiogram.types.passport_data, 345
aiogram.types.input_media_animation, 183
                                               aiogram.types.passport_element_error, 346
aiogram.types.input_media_audio, 184
                                               aiogram.types.passport_element_error_data_field,
aiogram.types.input_media_document, 186
aiogram.types.input_media_photo, 187
                                               aiogram.types.passport_element_error_file,
aiogram.types.input_media_video, 187
aiogram.types.input_message_content, 323
                                               aiogram.types.passport_element_error_files,
aiogram.types.input_paid_media, 189
aiogram.types.input_paid_media_photo, 189
                                               aiogram.types.passport_element_error_front_side,
aiogram.types.input_paid_media_video, 190
aiogram.types.input_poll_option, 191
                                               aiogram.types.passport_element_error_reverse_side,
aiogram.types.input_sticker, 340
aiogram.types.input_text_message_content, 323
                                               aiogram.types.passport_element_error_selfie,
aiogram.types.input_venue_message_content,
                                               aiogram.types.passport_element_error_translation_file,
aiogram.types.invoice, 326
aiogram.types.keyboard_button, 191
                                               aiogram.types.passport_element_error_translation_files,
aiogram.types.keyboard_button_poll_type, 192
                                                       351
aiogram.types.keyboard_button_request_chat,
```

714 Python Module Index

```
aiogram.types.passport_element_error_unspecifiæibgram.types.user, 270
        351
                                               aiogram.types.user_chat_boosts, 272
                                               aiogram.types.user_profile_photos, 272
aiogram.types.passport_file, 352
aiogram.types.photo_size, 262
                                               aiogram.types.user_shared, 272
aiogram.types.poll, 262
                                               aiogram.types.users_shared, 273
aiogram.types.poll_answer, 263
                                               aiogram.types.venue, 273
aiogram.types.poll_option, 264
                                               aiogram.types.video, 274
                                               aiogram.types.video_chat_ended, 275
aiogram.types.pre_checkout_query, 327
aiogram.types.prepared_inline_message, 325
                                               aiogram.types.video_chat_participants_invited,
aiogram.types.proximity_alert_triggered, 264
aiogram.types.reaction_count, 264
                                               aiogram.types.video_chat_scheduled, 275
                                               aiogram.types.video_chat_started, 275
aiogram.types.reaction_type, 265
aiogram.types.reaction_type_custom_emoji, 265
                                               aiogram.types.video_note, 276
aiogram.types.reaction_type_emoji, 265
                                               aiogram.types.voice, 276
aiogram.types.reaction_type_paid, 266
                                               aiogram.types.web_app_data, 277
aiogram.types.refunded_payment, 328
                                               aiogram.types.web_app_info, 277
aiogram.types.reply_keyboard_markup, 266
                                               aiogram.types.webhook_info, 355
aiogram.types.reply_keyboard_remove, 267
                                               aiogram.types.write_access_allowed, 277
aiogram.types.reply_parameters, 267
aiogram.types.response_parameters, 268
aiogram.types.revenue_withdrawal_state, 329
aiogram.types.revenue_withdrawal_state_failed,
        329
aiogram.types.revenue_withdrawal_state_pending,
aiogram.types.revenue_withdrawal_state_succeeded,
aiogram.types.sent_web_app_message, 325
aiogram.types.shared_user, 268
aiogram.types.shipping_address, 331
aiogram.types.shipping_option, 331
aiogram.types.shipping_query, 332
aiogram.types.star_transaction, 333
aiogram.types.star_transactions, 334
aiogram.types.sticker, 341
aiogram.types.sticker_set, 343
aiogram.types.story, 269
aiogram.types.successful_payment, 334
aiogram.types.switch_inline_query_chosen_chat,
        269
aiogram.types.text_quote, 270
aiogram.types.transaction_partner, 335
aiogram.types.transaction_partner_affiliate_program,
        335
aiogram.types.transaction_partner_chat, 336
aiogram.types.transaction_partner_fragment,
aiogram.types.transaction_partner_other, 337
aiogram.types.transaction_partner_telegram_ads,
aiogram.types.transaction_partner_telegram_api,
        338
aiogram.types.transaction_partner_user, 339
aiogram.types.update, 353
```

Python Module Index 715

716 Python Module Index

INDEX

Symbols method), 622
call() (aiogram.dispatcher.middlewares.base.BaseMiddleware () (aiogram.webhook.aiohttp_server.TokenBasedRequestHandler method), 615 method), 623
call() (aiogram.filters.base.Filter method), 582
method), 565init() (aiogram.dispatcher.router.Router method), A
accent_color_id (aiogram.types.chat.Chat attribute),
init() (aiogram.filters.command.Command 33 method), 571 accent_color_id (aiogram.types.chat_full_info.ChatFullInfo
init() (aiogram.fsm.storage.memory.MemoryStorage attribute), 54
method), 592 action (aiogram.methods.send_chat_action.SendChatActioninit() (aiogram.fsm.storage.mongo.MongoStorage attribute), 438
method), 593 actions (aiogram.fsm.scene.SceneConfig attribute), 611
init() (aiogram.fsm.storage.redis.RedisStorage active_usernames (aiogram.types.chat.Chat attribute), method), 592 34
init() (aiogram.types.input_file.BufferedInputFile active_usernames(aiogram.types.chat_full_info.ChatFullInfo attribute), 55
init() (aiogram.types.input_file.FSInputFile actor_chat(aiogram.types.message_reaction_updated.MessageReactionL
init() (aiogram.utils.callback_answer.CallbackAnswedd() (aiogram.fsm.scene.SceneRegistry method), 609
method), 651 add() (aiogram.utils.keyboard.InlineKeyboardBuilder
init() (aiogram.utils.callback_answer.CallbackAnswerMiddlewnethod), 636 method), 650 add() (aiogram.utils.keyboard.ReplyKeyboardBuilder
init() (aiogram.utils.chat_action.ChatActionSender method), 638
method), 644 add() (aiogram.utils.media_group.MediaGroupBuilderinit() (aiogram.utils.formatting.Text method), 655 method), 659
init() (aiogram.utils.i18n.middleware.ConstI18nMid atld waredio() (aiogram.utils.media_group.MediaGroupBuilder
method), 641 method), 659init() (aiogram.utils.i18n.middleware.FSMI18nMiddledadate (aiogram.types.chat_boost.ChatBoost at-
method), 641 tribute), 50
init() (aiogram.utils.i18n.middleware.I18nMiddlewa re ld_document() (aiogram.utils.media_group.MediaGroupBuilder method), 642 method), 660
init() (aiogram.utils.i18n.middleware.SimpleI18nMi ddl ew pho to() (aiogram.utils.media_group.MediaGroupBuilder
init() (aiogram.utils.keyboard.InlineKeyboardBuilderdd_to_router() (aiogram.fsm.scene.Scene class
method), 636 method), 609init() (aiogram.utils.keyboard.ReplyKeyboardBuildeadd_video() (aiogram.utils.media_group.MediaGroupBuilder
method), 638 method), 661
init() (aiogram.webhook.aiohttp_server.BaseReques#ddadleto_attachment_menu (aiogram.types.user.User method), 622 attribute), 271
init() (aiogram.webhook.aiohttp_server.SimpleRequ edtRe dato_attachment_menu

(aiogram.utils.web_app.WebAppUser at- tribute), 648	aiogram.enums.dice_emoji module,544
additional_chat_count (aiogram.types.giveaway_winners.GiveawayWinn	
attribute), 138	<pre>aiogram.enums.inline_query_result_type</pre>
${\tt ADDRESS}\ (a io gram.enums.encrypted_passport_element. Encoder and a passport_element. En$	crypte dBdxxpe rt El ement
attribute), 545	aiogram.enums.input_media_type
address (aiogram.methods.send_venue.SendVenue at-	module, 546
tribute), 463	<pre>aiogram.enums.input_paid_media_type</pre>
$address ({\it aiogram.types.business_location.BusinessLocations}) and {\it aiogram.types.business_location.BusinessLocations} and {\it aiogram.types.business_locations} and {\it aiogram.t$	
attribute), 30 address (aiogram.types.chat_location.ChatLocation at-	<pre>aiogram.enums.keyboard_button_poll_type_type module,547</pre>
tribute), 100	aiogram.enums.mask_position_point
address (aiogram.types.inline_query_result_venue.InlineQ	
attribute), 313	aiogram.enums.menu_button_type
address (aiogram.types.input_venue_message_content.Inp	
attribute), 324	aiogram.enums.message_entity_type
address (aiogram.types.venue.Venue attribute), 273	module, 548
	aiogram.enums.message_origin_type
aiogram.methods.add_sticker_to_set), 357	module, 548
adjust() (aiogram.utils.keyboard.InlineKeyboardBuilder	
method), 637	module, 549
adjust() (aiogram.utils.keyboard.ReplyKeyboardBuilder	
method), 638	module, 549
ADMINISTRATOR (aiogram.enums.chat_member_status.Cha	
attribute), 539	module, 549
AED (aiogram.enums.currency.Currency attribute), 541	aiogram.enums.poll_type
affiliate (aiogram.types.transaction_partner_user.Trans	
attribute), 339	aiogram.enums.reaction_type_type
affiliate_chat (aiogram.types.affiliate_info.AffiliateInfo	
attribute), 325	aiogram.enums.revenue_withdrawal_state_type
AFFILIATE_PROGRAM(aiogram.enums.transaction_partner	
attribute), 552	aiogram.enums.sticker_format
affiliate_user (aiogram.types.affiliate_info.AffiliateInfo	
attribute), 325	aiogram.enums.sticker_type
AffiliateInfo (class in aiogram.types.affiliate_info),	module, 551
325	aiogram.enums.topic_icon_color
AFN (aiogram.enums.currency.Currency attribute), 541	module, 551
aiogram.dispatcher.flags	aiogram.enums.transaction_partner_type
module, 620	module, 552
aiogram.enums.bot_command_scope_type	aiogram.enums.update_type
module, 537	module, 552
aiogram.enums.chat_action	aiogram.exceptions
module, 538	module, 618
aiogram.enums.chat_boost_source_type	aiogram.handlers.callback_query
module, 538	module, 631
aiogram.enums.chat_member_status	aiogram.methods.add_sticker_to_set
module, 539	module, 357
aiogram.enums.chat_type	aiogram.methods.answer_callback_query
module, 539	module, 379
aiogram.enums.content_type	aiogram.methods.answer_inline_query
module, 539	module, 510
aiogram.enums.currency	aiogram.methods.answer_pre_checkout_query
module, 541	module, 520
 	

aiogram.methods.answer_shipping_query module, 521	<pre>aiogram.methods.edit_forum_topic module, 402</pre>
<pre>aiogram.methods.answer_web_app_query module,512</pre>	<pre>aiogram.methods.edit_general_forum_topic module, 404</pre>
<pre>aiogram.methods.approve_chat_join_request module, 381</pre>	aiogram.methods.edit_message_caption module, 499
aiogram.methods.ban_chat_member module,382	aiogram.methods.edit_message_live_location module, 500
aiogram.methods.ban_chat_sender_chat module, 383	aiogram.methods.edit_message_media module, 503
aiogram.methods.close module, 384	<pre>aiogram.methods.edit_message_reply_markup module,504</pre>
<pre>aiogram.methods.close_forum_topic module, 385</pre>	<pre>aiogram.methods.edit_message_text module, 505</pre>
<pre>aiogram.methods.close_general_forum_topic module, 386</pre>	<pre>aiogram.methods.edit_user_star_subscription module, 525</pre>
<pre>aiogram.methods.copy_message module, 387</pre>	<pre>aiogram.methods.export_chat_invite_link module, 405</pre>
aiogram.methods.copy_messages module, 389	aiogram.methods.forward_message module, 406
<pre>aiogram.methods.create_chat_invite_link module, 391</pre>	aiogram.methods.forward_messages module,407
$\begin{tabular}{ll} {\tt aiogram.methods.create_chat_subscription_invi}\\ {\tt module}, 392 \end{tabular}$	tæidgirdæm.methods.get_available_gifts module,361
<pre>aiogram.methods.create_forum_topic module, 393</pre>	aiogram.methods.get_business_connection module, 408
<pre>aiogram.methods.create_invoice_link module, 523</pre>	aiogram.methods.get_chat module,409
<pre>aiogram.methods.create_new_sticker_set module, 358</pre>	<pre>aiogram.methods.get_chat_administrators module, 410</pre>
<pre>aiogram.methods.decline_chat_join_request module, 395</pre>	<pre>aiogram.methods.get_chat_member module, 411</pre>
<pre>aiogram.methods.delete_chat_photo module, 396</pre>	<pre>aiogram.methods.get_chat_member_count module, 412</pre>
<pre>aiogram.methods.delete_chat_sticker_set module, 397</pre>	<pre>aiogram.methods.get_chat_menu_button module, 412</pre>
<pre>aiogram.methods.delete_forum_topic module, 398</pre>	<pre>aiogram.methods.get_custom_emoji_stickers module, 362</pre>
aiogram.methods.delete_message module,496	<pre>aiogram.methods.get_file module, 413</pre>
aiogram.methods.delete_messages module, 498	<pre>aiogram.methods.get_forum_topic_icon_stickers module,414</pre>
<pre>aiogram.methods.delete_my_commands module, 399</pre>	<pre>aiogram.methods.get_game_high_scores module,516</pre>
<pre>aiogram.methods.delete_sticker_from_set module, 359</pre>	aiogram.methods.get_me module,415
<pre>aiogram.methods.delete_sticker_set module, 360</pre>	<pre>aiogram.methods.get_my_commands module, 415</pre>
<pre>aiogram.methods.delete_webhook module, 531</pre>	<pre>aiogram.methods.get_my_default_administrator_rights module, 416</pre>
<pre>aiogram.methods.edit_chat_invite_link module, 400</pre>	<pre>aiogram.methods.get_my_description module, 417</pre>
aiogram.methods.edit_chat_subscription_invite	_diiongram.methods.get_my_name module 418
mean 1 C 401	manute 410

<pre>aiogram.methods.get_my_short_description module, 419</pre>	<pre>aiogram.methods.send_game module, 517</pre>
<pre>aiogram.methods.get_star_transactions module, 526</pre>	<pre>aiogram.methods.send_gift module, 366</pre>
aiogram.methods.get_sticker_set	aiogram.methods.send_invoice
module, 363	module, 528
aiogram.methods.get_updates	aiogram.methods.send_location
module, 532	module, 446
<pre>aiogram.methods.get_user_chat_boosts module, 419</pre>	<pre>aiogram.methods.send_media_group module, 449</pre>
<pre>aiogram.methods.get_user_profile_photos module, 420</pre>	<pre>aiogram.methods.send_message module, 451</pre>
<pre>aiogram.methods.get_webhook_info module, 533</pre>	aiogram.methods.send_paid_media module,454
aiogram.methods.hide_general_forum_topic	aiogram.methods.send_photo
module, 421	module, 456
aiogram.methods.leave_chat	aiogram.methods.send_poll
module, 422	module, 459
aiogram.methods.log_out	aiogram.methods.send_sticker
module, 423	module, 367
aiogram.methods.pin_chat_message	aiogram.methods.send_venue
module, 424	module, 463
aiogram.methods.promote_chat_member	aiogram.methods.send_video
module, 425	module, 465
aiogram.methods.refund_star_payment	aiogram.methods.send_video_note
module, 527	module, 469
aiogram.methods.remove_chat_verification	aiogram.methods.send_voice
module, 363	module, 471
aiogram.methods.remove_user_verification	aiogram.methods.set_chat_administrator_custom_title
module, 364	module, 474
aiogram.methods.reopen_forum_topic module, 427	<pre>aiogram.methods.set_chat_description module, 475</pre>
<pre>aiogram.methods.reopen_general_forum_topic</pre>	aiogram.methods.set_chat_menu_button
module, 428	module, 476
<pre>aiogram.methods.replace_sticker_in_set module, 365</pre>	<pre>aiogram.methods.set_chat_permissions module, 477</pre>
<pre>aiogram.methods.restrict_chat_member</pre>	aiogram.methods.set_chat_photo
module, 429	module, 478
<pre>aiogram.methods.revoke_chat_invite_link module, 430</pre>	<pre>aiogram.methods.set_chat_sticker_set module, 479</pre>
<pre>aiogram.methods.save_prepared_inline_message</pre>	aiogram.methods.set_chat_title
module, 513	module, 480
aiogram.methods.send_animation	$\verb"aiogram.methods.set_custom_emoji_sticker_set_thumbnail"$
module, 432	module, 370
aiogram.methods.send_audio	aiogram.methods.set_game_score
module, 435	module, 519
<pre>aiogram.methods.send_chat_action module, 437</pre>	aiogram.methods.set_message_reaction module,481
aiogram.methods.send_contact	aiogram.methods.set_my_commands
module, 439	module, 483
aiogram.methods.send_dice	<pre>aiogram.methods.set_my_default_administrator_rights</pre>
module, 441	module, 484
aiogram.methods.send_document	aiogram.methods.set_my_description
module, 443	module, 485

<pre>aiogram.methods.set_my_name module, 486</pre>	<pre>aiogram.types.background_fill_freeform_gradient module, 21</pre>
aiogram.methods.set_my_short_description module, 487	<pre>aiogram.types.background_fill_gradient module, 21</pre>
aiogram.methods.set_passport_data_errors module,536	aiogram.types.background_fill_solid module, 22
aiogram.methods.set_sticker_emoji_list module, 371	aiogram.types.background_type module, 22
aiogram.methods.set_sticker_keywords module,372	<pre>aiogram.types.background_type_chat_theme module, 22</pre>
<pre>aiogram.methods.set_sticker_mask_position module, 373</pre>	<pre>aiogram.types.background_type_fill module, 23</pre>
<pre>aiogram.methods.set_sticker_position_in_set module, 374</pre>	<pre>aiogram.types.background_type_pattern module, 23</pre>
<pre>aiogram.methods.set_sticker_set_thumbnail module, 375</pre>	<pre>aiogram.types.background_type_wallpaper module, 24</pre>
<pre>aiogram.methods.set_sticker_set_title module, 376</pre>	aiogram.types.birthdate module, 24
<pre>aiogram.methods.set_user_emoji_status module, 488</pre>	<pre>aiogram.types.bot_command module, 25</pre>
<pre>aiogram.methods.set_webhook module, 534</pre>	<pre>aiogram.types.bot_command_scope module, 25</pre>
<pre>aiogram.methods.stop_message_live_location module, 507</pre>	<pre>aiogram.types.bot_command_scope_all_chat_administrators module, 25</pre>
<pre>aiogram.methods.stop_poll module, 509</pre>	<pre>aiogram.types.bot_command_scope_all_group_chats module, 26</pre>
<pre>aiogram.methods.unban_chat_member module, 489</pre>	<pre>aiogram.types.bot_command_scope_all_private_chats module, 26</pre>
<pre>aiogram.methods.unban_chat_sender_chat module, 490</pre>	<pre>aiogram.types.bot_command_scope_chat module, 26</pre>
<pre>aiogram.methods.unhide_general_forum_topic module, 491</pre>	<pre>aiogram.types.bot_command_scope_chat_administrators module, 27</pre>
<pre>aiogram.methods.unpin_all_chat_messages module, 492</pre>	<pre>aiogram.types.bot_command_scope_chat_member module, 27</pre>
${\tt aiogram.methods.unpin_all_forum_topic_message} \\ {\tt module}, 493$	esaiogram.types.bot_command_scope_default module,28
<pre>aiogram.methods.unpin_all_general_forum_topic module, 494</pre>	c_micosgraguestypes.bot_description module, 28
aiogram.methods.unpin_chat_message module, 495	aiogram.types.bot_name module,28
<pre>aiogram.methods.upload_sticker_file module, 377</pre>	<pre>aiogram.types.bot_short_description module, 28</pre>
aiogram.methods.verify_chat module, 378	aiogram.types.business_connection module, 29
<pre>aiogram.methods.verify_user module, 379</pre>	aiogram.types.business_intro module, 29
<pre>aiogram.types.affiliate_info module, 325</pre>	<pre>aiogram.types.business_location module, 30</pre>
aiogram.types.animation module, 19	<pre>aiogram.types.business_messages_deleted module, 30</pre>
aiogram.types.audio module, 20	aiogram.types.business_opening_hours module, 30
<pre>aiogram.types.background_fill module, 21</pre>	<pre>aiogram.types.business_opening_hours_interval module, 31</pre>

aiogram.types.callback_game module,356	<pre>aiogram.types.chat_shared module, 129</pre>
aiogram.types.callback_query module,31	<pre>aiogram.types.chosen_inline_result module, 278</pre>
aiogram.types.chat module,32	aiogram.types.contact module,129
aiogram.types.chat_administrator_rights module,48	<pre>aiogram.types.copy_text_button module, 130</pre>
aiogram.types.chat_background module,50	aiogram.types.dice module,130
aiogram.types.chat_boost module,50	<pre>aiogram.types.document module, 130</pre>
<pre>aiogram.types.chat_boost_added module, 50</pre>	<pre>aiogram.types.encrypted_credentials module, 344</pre>
<pre>aiogram.types.chat_boost_removed module, 51</pre>	<pre>aiogram.types.encrypted_passport_element module, 344</pre>
<pre>aiogram.types.chat_boost_source module, 51</pre>	<pre>aiogram.types.error_event module,617</pre>
<pre>aiogram.types.chat_boost_source_gift_code module, 51</pre>	<pre>aiogram.types.external_reply_info module, 131</pre>
<pre>aiogram.types.chat_boost_source_giveaway module, 52</pre>	aiogram.types.file module,133
<pre>aiogram.types.chat_boost_source_premium module, 52</pre>	<pre>aiogram.types.force_reply module, 133</pre>
<pre>aiogram.types.chat_boost_updated module, 53</pre>	<pre>aiogram.types.forum_topic module, 134</pre>
<pre>aiogram.types.chat_full_info module, 53</pre>	<pre>aiogram.types.forum_topic_closed module, 134</pre>
<pre>aiogram.types.chat_invite_link module, 57</pre>	<pre>aiogram.types.forum_topic_created module, 135</pre>
aiogram.types.chat_join_request module,58	<pre>aiogram.types.forum_topic_edited module, 135</pre>
aiogram.types.chat_location module,100	<pre>aiogram.types.forum_topic_reopened module, 135</pre>
aiogram.types.chat_member module,100	aiogram.types.game module,356
aiogram.types.chat_member_administrator module,101	<pre>aiogram.types.game_high_score module, 357</pre>
aiogram.types.chat_member_banned module,102	<pre>aiogram.types.general_forum_topic_hidden module, 135</pre>
aiogram.types.chat_member_left module,103	<pre>aiogram.types.general_forum_topic_unhidden module, 136</pre>
aiogram.types.chat_member_member module,103	aiogram.types.gift module,340
aiogram.types.chat_member_owner module,104	aiogram.types.gifts module,340
aiogram.types.chat_member_restricted module, 104	aiogram.types.giveaway module,136
aiogram.types.chat_member_updated module, 106	<pre>aiogram.types.giveaway_completed module, 137</pre>
aiogram.types.chat_permissions module,127	<pre>aiogram.types.giveaway_created module, 137</pre>
aiogram.types.chat_photo module,128	<pre>aiogram.types.giveaway_winners module, 137</pre>

```
aiogram.types.inaccessible_message
                                               aiogram.types.input_file
                                                   module, 182
   module, 138
aiogram.types.inline_keyboard_button
                                               aiogram.types.input_invoice_message_content
   module, 181
                                                   module, 319
aiogram.types.inline_keyboard_markup
                                               aiogram.types.input_location_message_content
   module, 182
                                                   module, 322
aiogram.types.inline_query
                                               aiogram.types.input_media
    module, 278
                                                   module, 183
aiogram.types.inline_query_result
                                               aiogram.types.input_media_animation
    module, 279
                                                   module, 183
aiogram.types.inline_query_result_article
                                               aiogram.types.input_media_audio
                                                   module, 184
    module, 280
aiogram.types.inline_query_result_audio
                                               aiogram.types.input_media_document
    module, 282
                                                   module, 186
aiogram.types.inline_query_result_cached_audioaiogram.types.input_media_photo
    module, 283
                                                   module, 187
aiogram.types.inline_query_result_cached_documaendgram.types.input_media_video
                                                   module, 187
aiogram.types.inline_query_result_cached_gif aiogram.types.input_message_content
    module, 288
                                                   module, 323
aiogram.types.inline_query_result_cached_mpeg4aigigfram.types.input_paid_media
                                                   module, 189
    module, 289
aiogram.types.inline_query_result_cached_photoaiogram.types.input_paid_media_photo
    module, 292
                                                   module, 189
aiogram.types.inline_query_result_cached_stickæiogram.types.input_paid_media_video
    module, 294
                                                   module, 190
aiogram.types.inline_query_result_cached_videoaiogram.types.input_poll_option
    module, 296
                                                   module, 191
aiogram.types.inline_query_result_cached_voiceaiogram.types.input_sticker
    module, 298
                                                   module, 340
aiogram.types.inline_query_result_contact
                                               aiogram.types.input_text_message_content
    module, 300
                                                   module, 323
aiogram.types.inline_query_result_document
                                               aiogram.types.input_venue_message_content
    module, 301
                                                   module, 324
aiogram.types.inline_query_result_game
                                               aiogram.types.invoice
   module, 303
                                                   module, 326
aiogram.types.inline_query_result_gif
                                               aiogram.types.keyboard_button
    module, 304
                                                   module, 191
aiogram.types.inline_query_result_location
                                               aiogram.types.keyboard_button_poll_type
   module, 305
                                                   module, 192
                                               aiogram.types.keyboard_button_request_chat
aiogram.types.inline_query_result_mpeg4_gif
    module, 308
                                                   module, 192
aiogram.types.inline_query_result_photo
                                               aiogram.types.keyboard_button_request_user
    module, 311
                                                   module, 194
aiogram.types.inline_query_result_venue
                                               aiogram.types.keyboard_button_request_users
    module, 313
                                                   module, 195
aiogram.types.inline_query_result_video
                                               aiogram.types.labeled_price
    module, 314
                                                   module, 326
aiogram.types.inline_query_result_voice
                                               aiogram.types.link_preview_options
    module, 316
                                                   module, 196
aiogram.types.inline_query_results_button
                                               aiogram.types.location
    module, 318
                                                   module, 196
aiogram.types.input_contact_message_content
                                               aiogram.types.login_url
    module, 318
                                                   module, 197
```

aiogram turnos mask nosition	siggram types passport alament orror file
aiogram.types.mask_position module, 341	<pre>aiogram.types.passport_element_error_file module, 347</pre>
aiogram.types.maybe_inaccessible_message module, 198	<pre>aiogram.types.passport_element_error_files module, 347</pre>
aiogram.types.menu_button module,198	<pre>aiogram.types.passport_element_error_front_side module, 348</pre>
aiogram.types.menu_button_commands	aiogram.types.passport_element_error_reverse_side
module, 198	module, 349
aiogram.types.menu_button_default module, 199	<pre>aiogram.types.passport_element_error_selfie module, 349</pre>
<pre>aiogram.types.menu_button_web_app module, 199</pre>	<pre>aiogram.types.passport_element_error_translation_file module, 350</pre>
aiogram.types.message module, 199	<pre>aiogram.types.passport_element_error_translation_files module, 351</pre>
	extiogram.types.passport_element_error_unspecified module, 351
aiogram.types.message_entity	aiogram.types.passport_file
module, 255	module, 352
<pre>aiogram.types.message_id module, 256</pre>	aiogram.types.photo_size module,262
aiogram.types.message_origin	aiogram.types.poll
module, 256	module, 262
<pre>aiogram.types.message_origin_channel module, 257</pre>	aiogram.types.poll_answer module,263
aiogram.types.message_origin_chat	aiogram.types.poll_option
module, 257	module, 264
aiogram.types.message_origin_hidden_user module, 258	<pre>aiogram.types.pre_checkout_query module, 327</pre>
aiogram.types.message_origin_user module, 258	<pre>aiogram.types.prepared_inline_message module, 325</pre>
<pre>aiogram.types.message_reaction_count_updated module, 259</pre>	aiogram.types.proximity_alert_triggered module, 264
aiogram.types.message_reaction_updated	aiogram.types.reaction_count
module, 259	module, 264
aiogram.types.order_info	aiogram.types.reaction_type
module, 327	module, 265
aiogram.types.paid_media module, 260	aiogram.types.reaction_type_custom_emoji module,265
aiogram.types.paid_media_info module, 260	aiogram.types.reaction_type_emoji module,265
<pre>aiogram.types.paid_media_photo module, 261</pre>	aiogram.types.reaction_type_paid module, 266
<pre>aiogram.types.paid_media_preview module, 261</pre>	<pre>aiogram.types.refunded_payment module, 328</pre>
<pre>aiogram.types.paid_media_purchased module, 327</pre>	aiogram.types.reply_keyboard_markup module,266
aiogram.types.paid_media_video module, 261	aiogram.types.reply_keyboard_remove module,267
aiogram.types.passport_data	aiogram.types.reply_parameters
module, 345	module, 267
<pre>aiogram.types.passport_element_error module, 346</pre>	aiogram.types.response_parameters module, 268
aiogram.types.passport_element_error_data_fie	
module 346	module, 329

<pre>aiogram.types.revenue_withdrawal_state_failed module, 329</pre>	aiogram.types.user_profile_photos module,272
aiogram.types.revenue_withdrawal_state_pendin module,330	
aiogram.types.revenue_withdrawal_state_succee module, 330	
aiogram.types.sent_web_app_message module, 325	aiogram.types.venue module, 273
aiogram.types.shared_user module, 268	aiogram.types.video module, 274
aiogram.types.shipping_address module,331	aiogram.types.video_chat_ended module, 275
aiogram.types.shipping_option module, 331	aiogram.types.video_chat_participants_invited module, 275
aiogram.types.shipping_query module,332	aiogram.types.video_chat_scheduled module, 275
aiogram.types.star_transaction module, 333	aiogram.types.video_chat_started module, 275
aiogram.types.star_transactions module, 334	aiogram.types.video_note module, 276
aiogram.types.sticker module, 341	aiogram.types.voice module, 276
aiogram.types.sticker_set module, 343	aiogram.types.web_app_data module, 277
aiogram.types.story module, 269	aiogram.types.web_app_info module, 277
aiogram.types.successful_payment module, 334	aiogram.types.webhook_info module, 355
aiogram.types.switch_inline_query_chosen_chat module, 269	
aiogram.types.text_quote module, 270	AiogramError, 618 AiohttpSession (class in
aiogram.types.transaction_partner module, 335	aiogram.client.session.aiohttp), 16 ALL (aiogram.enums.currency.Currency attribute), 541
aiogram.types.transaction_partner_affiliate_p module,335	
aiogram.types.transaction_partner_chat module, 336	attribute), 537 ALL_GROUP_CHATS (aiogram.enums.bot_command_scope_type.BotComma
aiogram.types.transaction_partner_fragment module, 337	attribute), 537 ALL_PRIVATE_CHATS (aiogram.enums.bot_command_scope_type.BotCommand_scop
aiogram.types.transaction_partner_other module, 337	attribute), 537 allow_bot_chats (aiogram.methods.save_prepared_inline_message.Save
aiogram.types.transaction_partner_telegram_ad module, 338	, o — — — — — — — — — — — — — — — — — —
aiogram.types.transaction_partner_telegram_ap module,338	
aiogram.types.transaction_partner_user module, 339	(aiogram.methods.save_prepared_inline_message.SavePreparedlattribute), 515
aiogram.types.update module, 353	allow_channel_chats (aiogram.types.switch_inline_query_chosen_chat.SwitchInlineQu
aiogram.types.user	attribute), 270 allow group chats (giogram methods save prepared inline message Sa

aiogram.types.user_chat_boosts

module, 272

attribute), 515

allow_group_chats (aiogram.types.switch_inline_query_attribute), 269	_chosen_ch (ati&gniuch.lmkitha@n.exeyld<u>h</u>vixlenC_lna tte.SendVideoNote attribute), 470
allow_paid_broadcast	allow_paid_broadcast
(aiogram.methods.copy_message.CopyMessage attribute), 388	(aiogram.methods.send_voice.SendVoice attribute), 472
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_animation.SendAnimatio attribute), 433	
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_audio.SendAudio attribute), 436	(aiogram.methods.copy_message.CopyMessage attribute), 388
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_contact.SendContact attribute), 440	(aiogram.methods.send_animation.SendAnimation attribute), 433
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_dice.SendDice at- tribute), 442	(aiogram.methods.send_audio.SendAudio attribute), 436
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_document.SendDocumen attribute), 445	at (aiogram.methods.send_contact.SendContact attribute), 440
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_game.SendGame attribute), 517	(aiogram.methods.send_dice.SendDice at- tribute), 442
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_invoice.SendInvoice attribute), 530	(aiogram.methods.send_document.SendDocument attribute), 445
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_location.SendLocation attribute), 448	(aiogram.methods.send_game.SendGame attribute), 518
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_media_group.SendMediaattribute), 450	aGroup (aiogram.methods.send_invoice.SendInvoice attribute), 530
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_message.SendMessage attribute), 453	(aiogram.methods.send_location.SendLocation attribute), 448
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_paid_media.SendPaidMeattribute), 455	attribute), 450
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_photo.SendPhoto attribute), 458	(aiogram.methods.send_message.SendMessage attribute), 453
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_poll.SendPoll at- tribute), 461	(aiogram.methods.send_photo.SendPhoto attribute), 458
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_sticker.SendSticker attribute), 368	(aiogram.methods.send_poll.SendPoll at- tribute), 461
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_venue.SendVenue attribute), 464	(aiogram.methods.send_sticker.SendSticker attribute), 369
allow_paid_broadcast	allow_sending_without_reply
(aiogram.methods.send_video.SendVideo attribute), 467	(aiogram.methods.send_venue.SendVenue attribute), 464
allow_paid_broadcast	allow_sending_without_reply

```
method), 106
                  (aiogram.methods.send_video.SendVideo
                  attribute), 468
                                                                                                              answer() (aiogram.types.inaccessible_message.InaccessibleMessage
allow_sending_without_reply
                                                                                                                                 method), 138
                  (aiogram.methods.send_video_note.SendVideoNotanswer()
                                                                                                                                              (aiogram.types.inline_query.InlineQuery
                  attribute), 470
                                                                                                                                 method), 279
allow_sending_without_reply
                                                                                                               answer() (aiogram.types.message.Message method),
                  (aiogram.methods.send voice.SendVoice
                  attribute), 472
                                                                                                               answer() (aiogram.types.pre_checkout_query.PreCheckoutQuery
allow_sending_without_reply
                                                                                                                                 method), 328
                  (aiogram.types.reply_parameters.ReplyParameter.answer() (aiogram.types.shipping_query.ShippingQuery
                  attribute), 267
                                                                                                                                 method), 332
allow_user_chats(aiogram.methods.save_prepared_inlinan_swess_qan_SmatRepQ):ddlnkineMessag.chat_join_request.ChatJoinRequest
                  attribute), 515
                                                                                                                                 method), 61
allow_user_chats(aiogram.types.switch_inline_query_chanswerhans\mitthbulineQuiogy@haspec.khtt_member_updated.ChatMembe
                  attribute), 269
                                                                                                                                  method), 108
allowed\_updates (a iogram. methods. get\_updates. GetUpdates. Wer\_animation() (a iogram. types. in accessible\_message. In accessible Message. The control of the control o
                  attribute), 533
                                                                                                                                 method), 141
allowed_updates(aiogram.methods.set_webhook.SetWebhooksWer_animation() (aiogram.types.message.Message
                  attribute), 535
                                                                                                                                 method), 208
allowed_updates(aiogram.types.webhook_info.Webhooklanfiswer_animation_pm()
                  attribute), 356
                                                                                                                                  (aiogram.types.chat_join_request.ChatJoinRequest
allows_multiple_answers
                                                                                                                                 method), 63
                  (aiogram.methods.send\_poll.SendPoll
                                                                                                              answer_audio() (aiogram.types.chat_join_request.ChatJoinRequest
                  tribute), 461
                                                                                                                                 method), 64
allows_multiple_answers
                                                            (aiogram.types.poll.Poll answer_audio() (aiogram.types.chat_member_updated.ChatMemberUpd
                  attribute), 263
                                                                                                                                 method), 109
allows\_write\_to\_pm(aiogram.utils.web\_app.WebAppUsaenswer\_audio() (aiogram.types.inaccessible\_message.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.InaccessibleMessage.Ina
                  attribute), 648
                                                                                                                                 method), 143
                                                                                                               answer_audio()
AMD (aiogram.enums.currency.Currency attribute), 541
                                                                                                                                                            (aiogram.types.message.Message
                  (aiogram.types.affiliate_info.AffiliateInfo
amount
                                                                                                                                 method), 210
                  tribute), 325
                                                                                                               answer_audio_pm() (aiogram.types.chat_join_request.ChatJoinRequest
amount (aiogram.types.labeled_price.LabeledPrice at-
                                                                                                                                 method), 65
                  tribute), 326
                                                                                                              answer_contact() (aiogram.types.chat_join_request.ChatJoinRequest
amount \ (aiogram.types.star\_transaction.StarTransaction
                                                                                                                                 method), 67
                  attribute), 333
                                                                                                              answer_contact() (aiogram.types.chat_member_updated.ChatMemberU
ANIMATED (aiogram.enums.sticker_format.StickerFormat
                                                                                                                                 method), 110
                  attribute), 551
                                                                                                               answer_contact() (aiogram.types.inaccessible message.InaccessibleMes
ANIMATION
                          (aiogram.enums.content_type.ContentType
                                                                                                                                 method), 146
                  attribute), 539
                                                                                                               answer_contact()
                                                                                                                                                         (aiogram.types.message.Message
ANIMATION (aiogram.enums.input_media_type.InputMediaType
                                                                                                                                 method), 213
                  attribute), 546
                                                                                                               answer_contact_pm()
\verb"animation" (a iogram.methods.send\_animation. Send Animation
                                                                                                                                 (aiogram.types.chat_join_request.ChatJoinRequest
                  attribute), 432
                                                                                                                                 method), 68
animation(aiogram.types.external_reply_info.ExternalReplyster_dice() (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                 method), 86
                  attribute), 132
                                                                                                               answer_dice() (aiogram.types.chat_member_updated.ChatMemberUpdated.
animation (aiogram.types.game.Game attribute), 356
animation (aiogram.types.message.Message attribute),
                                                                                                                                 method), 120
                                                                                                               answer_dice() (aiogram.types.inaccessible_message.InaccessibleMessag
                  202
Animation (class in aiogram.types.animation), 19
                                                                                                                                 method), 165
answer() (aiogram.types.callback_query.CallbackQuery
                                                                                                              answer_dice()
                                                                                                                                                            (aiogram.types.message.Message
                  method), 32
                                                                                                                                 method), 233
answer() (aiogram.types.chat_join_request.ChatJoinRequeatnswer_dice_pm() (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                 method), 87
                  method), 59
answer() (aiogram.types.chat member updated.ChatMemlarstyberbutedcument() (aiogram.types.chat join request.ChatJoinRequest
```

```
method), 69
                                                                                                                                                                           answer_media_group_pm()
answer_document() (aiogram.types.chat_member_updated.ChatMethioglipdatques.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                        method), 80
                            method), 111
answer_document() (aiogram.types.inaccessible_messagealmaweaxsplalidLawedga()
                            method), 148
                                                                                                                                                                                                         (aiogram.types.inaccessible_message.InaccessibleMessage
answer_document() (aiogram.types.message.Message
                                                                                                                                                                                                        method), 179
                            method), 215
                                                                                                                                                                           answer_paid_media()
answer_document_pm()
                                                                                                                                                                                                        (aiogram.types.message.Message
                                                                                                                                                                                                                                                                                                                       method),
                            (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                         253
                            method), 70
                                                                                                                                                                            answer_photo() (aiogram.types.chat_join_request.ChatJoinRequest
answer_game() (aiogram.types.chat_join_request.ChatJoinRequest_method), 81
                                                                                                                                                                            answer_photo() (aiogram.types.chat_member_updated.ChatMemberUpdated.
                            method), 71
answer_game() (aiogram.types.chat_member_updated.ChatMemberbhedluveld), 117
                            method), 113
                                                                                                                                                                            answer_photo() (aiogram.types.inaccessible_message.InaccessibleMessa
answer_game() (aiogram.types.inaccessible_message.InaccessibleMmsethged), 160
                            method), 151
                                                                                                                                                                            answer_photo()
                                                                                                                                                                                                                                                 (aiogram.types.message.Message
                                                                                                                                                                                                        method), 228
answer_game()
                                                                     (aiogram.types.message.Message
                            method), 217
                                                                                                                                                                            answer_photo_pm() (aiogram.types.chat_join_request.ChatJoinRequest
answer_game_pm() (aiogram.types.chat_join_request.ChatJoinRequesethod), 82
                            method), 72
                                                                                                                                                                            answer_pm() (aiogram.types.chat_join_request.ChatJoinRequest
answer_invoice() (aiogram.types.chat_join_request.ChatJoinRequesethod), 60
                            method), 73
                                                                                                                                                                            answer_poll() (aiogram.types.chat_join_request.ChatJoinRequest
answer_invoice() (aiogram.types.chat_member_updated.ChatMembethbub)ated
                                                                                                                                                                            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
                            method), 113
answer_invoice() (aiogram.types.inaccessible_message.InaccessiblatMessage19
                            method), 152
                                                                                                                                                                            answer_poll() (aiogram.types.inaccessible_message.InaccessibleMessag
answer_invoice()
                                                                     (aiogram.types.message.Message
                                                                                                                                                                                                        method), 162
                            method), 219
                                                                                                                                                                            answer_poll()
                                                                                                                                                                                                                                                 (aiogram.types.message.Message
answer_invoice_pm()
                                                                                                                                                                                                        method), 230
                            (aiogram.types.chat_join_request.ChatJoinRequesanswer_poll_pm() (aiogram.types.chat_join_request.ChatJoinRequesa
                            method), 75
                                                                                                                                                                                                        method), 85
answer_location() (aiogram.types.chat_join_request.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowerequest.Chatlowereq
                            method), 77
                                                                                                                                                                                                        method), 88
answer_location() (aiogram.types.chat_member_updatedxChutMesthexNeptQutediogram.types.chat_member_updated.ChatMemberU
                            method), 115
                                                                                                                                                                                                        method), 121
answer_location() (aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_message.InaccessibleMessagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess@aiogram.types.inaccessible_messagealmswers.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkess.siblivdkes
                            method), 156
                                                                                                                                                                                                        method), 167
answer_location() (aiogram.types.message.Message answer_sticker()
                                                                                                                                                                                                                                                (aiogram.types.message.Message
                            method), 223
                                                                                                                                                                                                        method), 234
answer_location_pm()
                                                                                                                                                                           answer_sticker_pm()
                             (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                        (aiogram.types.chat_join_request.ChatJoinRequest
                            method), 78
                                                                                                                                                                                                        method), 89
answer_media_group()
                                                                                                                                                                            answer_venue() (aiogram.types.chat_join_request.ChatJoinRequest
                            (aiogram.types.chat_join_request.ChatJoinRequest
                                                                                                                                                                                                        method), 90
                                                                                                                                                                            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
                            method), 79
answer_media_group()
                                                                                                                                                                                                        method), 122
                            (aiogram.types.chat_member_updated.ChatMembanlspdatedenue() (aiogram.types.inaccessible_message.InaccessibleMessa
                            method), 117
                                                                                                                                                                                                        method), 169
answer_media_group()
                                                                                                                                                                            answer_venue()
                                                                                                                                                                                                                                                 (aiogram.types.message.Message
                             (aiogram.types.inaccessible_message.InaccessibleMessage_method), 236
                            method), 158
                                                                                                                                                                            answer_venue_pm() (aiogram.types.chat_join_request.ChatJoinRequest
answer_media_group()
                                                                                                                                                                                                        method), 91
                             (aiogram.types.message.Message
                                                                                                                                          method),
                                                                                                                                                                           answer_video() (aiogram.types.chat_join_request.ChatJoinRequest
                            224
                                                                                                                                                                                                        method), 92
```

answer_video() (aiogram.types.chat_member_updated.C method), 123	Chapphonba)Updagram.types.chat_join_request.ChatJoinRequest method), 58
<pre>answer_video() (aiogram.types.inaccessible_message.In</pre>	akppsoble@hastdgainRequest (class in
method), 171	aiogram.methods.approve_chat_join_request),
<pre>answer_video() (aiogram.types.message.Message</pre>	381
method), 239	args (aiogram.filters.command.CommandObject at-
<pre>answer_video_note()</pre>	tribute), 572
(aiogram.types.chat_join_request.ChatJoinReque	estRS (aiogram.enums.currency.Currency attribute), 541
method), 95	$\verb ARTICLE (a iogram.enums.in line_query_result_type.In lineQueryResultType ARTICLE (a iogram.enums.in lineQueryResultType ARTICLE (a iogram.enum.enums.in lineQueryResultType ARTICLE (a iogram.enum.enum.enum.enum.enum.enum.enum.enu$
<pre>answer_video_note()</pre>	attribute), 546
	basUpdatther() (aiogram.fsm.scene.Scene class method),
method), 125	609
<pre>answer_video_note()</pre>	as_html() (aiogram.utils.formatting.Text method), 655
	edseksergevalue() (in module aiogram.utils.formatting),
method), 174	654
<pre>answer_video_note()</pre>	as_kwargs() (aiogram.utils.formatting.Text method),
(aiogram.types.message.Message method),	655
242	as_line() (in module aiogram.utils.formatting), 652
<pre>answer_video_note_pm()</pre>	as_list() (in module aiogram.utils.formatting), 652
	esas_markdown() (aiogram.utils.formatting.Text method),
method), 96	655
answer_video_pm() (aiogram.types.chat_join_request.Cl	
method), 94	aiogram.utils.formatting), 653
<pre>answer_voice() (aiogram.types.chat_join_request.ChatJo method), 97</pre>	oùsRaupueked_section() (in module aiogram.utils.formatting), 653
answer_voice() (aiogram.types.chat_member_updated.C method), 126	Chas Mambert Epil dried t () (in module aiogram.utils.formatting), 653
<pre>answer_voice() (aiogram.types.inaccessible_message.In</pre>	
method), 176	aiogram.utils.formatting), 653
answer_voice() (aiogram.types.message.Message method), 244	
answer_voice_pm() (aiogram.types.chat_join_request.Cl	
method), 99	as_reply_parameters()
AnswerCallbackQuery (class in	(aiogram.types.message.Message method),
aiogram.methods.answer_callback_query),	206
379	as_router() (aiogram.fsm.scene.Scene class method),
answered (aiogram.utils.callback_answer.CallbackAnswer	
property), 651	as_section() (in module aiogram.utils.formatting), 653
AnswerInlineQuery (class in	attrs_resolver() (aiogram.fsm.scene.SceneConfig
aiogram.methods.answer_inline_query),	method), 611
510	AUD (aiogram.enums.currency.Currency attribute), 541
AnswerPreCheckoutQuery (class in	AUDIO (aiogram.enums.content_type.ContentType at-
aiogram.methods.answer_pre_checkout_query),	tribute), 539
520	AUDIO (aiogram.enums.inline_query_result_type.InlineQueryResultType
AnswerShippingQuery (class in	attribute), 545
aiogram.methods.answer_shipping_query),	AUDIO (aiogram.enums.input_media_type.InputMediaType
521	attribute), 546
AnswerWebAppQuery (class in	audio (aiogram.methods.send_audio.SendAudio at-
aiogram.methods.answer_web_app_query),	tribute), 435
512	audio (aiogram.types.external_reply_info.ExternalReplyInfo
ANY (aiogram.enums.content_type.ContentType at-	attribute), 132
tribute), 539	audio (aiogram.types.message.Message attribute), 202
<pre>api_url() (aiogram.client.telegram.TelegramAPIServer</pre>	Audio (class in aiogram.types.audio), 20
method), 14	<pre>audio_duration(aiogram.types.inline_query_result_audio.InlineQuery</pre>

attribute), 283	BanChatMember (class in
audio_file_id(aiogram.types.inline_query_result_cach	
attribute), 285	BanChatSenderChat (class in
audio_url (aiogram.types.inline_query_result_audio.Inlinattribute), 283	neQueryRe xuagnam ametnoas.ban_cnat_senaer_cnat), 383
auth_date (aiogram.utils.web_app.WebAppInitData attribute), 647	BANK_STATEMENT (aiogram.enums.encrypted_passport_element.Encrypted_attribute), 545
author_signature (aiogram.types.message.Message	base (aiogram.client.telegram.TelegramAPIServer
attribute), 202	attribute), 14
author_signature(aiogram.types.message_origin_chan attribute), 257	n BhshMideiQvigin Channel (class in aiogram.dispatcher.middlewares.base), 615
author_signature(aiogram.types.message_origin_chat.	MeaseReQuesttHandler (class in
attribute), 257	aiogram.webhook.aiohttp_server), 621
	BaseSession (class in aiogram.client.session.base), 15
attribute), 34	BaseStorage (class in aiogram.fsm.storage.base), 594
available_reactions	BASKETBALL (aiogram.enums.dice_emoji.DiceEmoji at-
(aiogram.types.chat_full_info.ChatFullInfo	tribute), 544
attribute), 55	BASKETBALL (aiogram.types.dice.DiceEmoji attribute),
AZN (aiogram.enums.currency.Currency attribute), 542	130
В	BDT (aiogram.enums.currency.Currency attribute), 542
	BGN (aiogram.enums.currency.Currency attribute), 542
back() (aiogram.fsm.scene.SceneWizard method), 611	big_file_id (aiogram.types.chat_photo.ChatPhoto at- tribute), 128
background_custom_emoji_id	big_file_unique_id(aiogram.types.chat_photo.ChatPhoto
(aiogram.types.chat.Chat attribute), 34	attribute), 129
background_custom_emoji_id (aiogram.types.chat_full_info.ChatFullInfo	bio (aiogram.types.chat.Chat attribute), 34
attribute), 55	bio (aiogram.types.chat_full_info.ChatFullInfo at-
BackgroundFill (class in	tribute), 56 bio (aiogram.types.chat_join_request.ChatJoinRequest
aiogram.types.background_fill), 21	attribute), 58
BackgroundFillFreeformGradient (class in	hthirthdate (aiogram.types.chat.Chat attribute), 34
aiogram.types.backgrouna_лиfreeform_gradien 21	birthdate (aiogram.types.chat_full_info.ChatFullInfo
	attribute), 55
BackgroundFillGradient (class in aiogram.types.background_fill_gradient),	Birthdate (class in aiogram.types.birthdate), 24
21	BLOCKQUOTE (aiogram.enums.message_entity_type.MessageEntityType
BackgroundFillSolid (class in	attribute), 548
aiogram.types.background_fill_solid), 22	BlockQuote (class in aiogram.utils.formatting), 658
BackgroundType (class in	BLUE (aiogram.enums.topic_icon_color.TopicIconColor
aiogram.types.background_type), 22	attribute), 551
BackgroundTypeChatTheme (class in	BND (aiogram.enums.currency.Currency attribute), 542
aiogram.types.background_type_chat_theme),	BOB (aiogram.enums.currency.Currency attribute), 542
22	BOLD (aiogram.enums.message_entity_type.MessageEntityType
BackgroundTypeFill (class in	attribute), 548
<pre>aiogram.types.background_type_fill), 23</pre>	Bold (class in aiogram.utils.formatting), 657
BackgroundTypePattern (class in	boost (aiogram.types.chat_boost_updated.ChatBoostUpdated
aiogram.types.background_type_pattern),	attribute), 53
23	BOOST_ADDED (aiogram.enums.content_type.ContentType
BackgroundTypeWallpaper (class in	attribute), 541
aiogram.types.background_type_wallpaper), 24	boost_added (aiogram.types.message.Message at- tribute), 205
BAM (aiogram.enums.currency.Currency attribute), 542	boost_count (aiogram.types.chat_boost_added.ChatBoostAdded
ban() (aiogram.types.chat.Chat method), 46	attribute), 50
ban_sender_chat() (aiogram.types.chat.Chat method),	boost_id (aiogram.types.chat_boost.ChatBoost at-
37	tribute), 50

```
boost_id(aiogram.types.chat boost removed.ChatBoostR#0WtANG(aiogram.types.dice.DiceEmoji attribute), 130
         attribute), 51
                                                      BRL (aiogram.enums.currency.Currency attribute), 542
boosts (aiogram.types.user chat boosts.UserChatBoosts BufferedInputFile (class in aiogram.types.input file),
                                                               182, 556
         attribute), 272
bot(aiogram.dispatcher.middlewares.data.MiddlewareDatdbuild()
                                                                (aiogram.fsm.storage.base.DefaultKeyBuilder
        attribute), 570
                                                               method), 594
bot_administrator_rights
                                                      build() (aiogram.fsm.storage.base.KeyBuilder method),
         (aiogram. types. keyboard\_button\_request\_chat. KeyboardBut \textit{Son} Request Chat
         attribute), 194
                                                      build() (aiogram.utils.media group.MediaGroupBuilder
BOT_COMMAND (aiogram.enums.message_entity_type.MessageEntityTyprethod), 662
         attribute), 548
                                                      BUSINESS_CONNECTION
bot_is_member(aiogram.types.keyboard_button_request_chat.Keyb@aid@www.nRurusestdluwt_type.UpdateType
                                                               attribute), 552
         attribute), 194
bot_username
                    (aiogram.types.login_url.LoginUrl
                                                      business_connection (aiogram.types.update.Update
         attribute), 197
                                                               attribute), 353
BotCommand (class in aiogram.types.bot_command), 25
                                                      business_connection_id
BotCommand (class in aiogram.utils.formatting), 656
                                                               (aiogram.methods.create_invoice_link.CreateInvoiceLink
BotCommandScope
                                                  in
                                                               attribute), 523
         aiogram.types.bot_command_scope), 25
                                                      business_connection_id
BotCommandScopeAllChatAdministrators (class in
                                                               (aiogram.methods.edit message caption.EditMessageCaption
        aiogram.types.bot_command_scope_all_chat_administratorattribute), 499
                                                      business_connection_id
BotCommandScopeAllGroupChats
                                                               (aiogram.methods.edit_message_live_location.EditMessageLiveL
                                       (class
                                                  in
         aiogram.types.bot command scope all group chats),
                                                               attribute), 501
                                                      business_connection_id
BotCommandScopeAllPrivateChats
                                                  in
                                                               (aiogram.methods.edit_message_media.EditMessageMedia
         aiogram.types.bot_command_scope_all_private_chats),
                                                               attribute), 503
                                                      business_connection_id
BotCommandScopeChat
                                                               (aiogram.methods.edit_message_reply_markup.EditMessageRepl
                                  (class
                                                  in
         aiogram.types.bot_command_scope_chat),
                                                               attribute), 504
                                                      business_connection_id
                                                               (aiogram.methods.edit\_message\_text.EditMessageText
BotCommandScopeChatAdministrators
                                         (class
                                                  in
        aiogram.types.bot_command_scope_chat_administrators), attribute), 506
                                                      business_connection_id
BotCommandScopeChatMember
                                                               (aiogram.methods.get business connection.GetBusinessConnecti
                                     (class
                                                  in
        aiogram.types.bot_command_scope_chat_member),
                                                               attribute), 408
         27
                                                      business_connection_id
BotCommandScopeDefault
                                   (class
                                                  in
                                                               (aiogram.methods.pin_chat_message.PinChatMessage
         aiogram.types.bot_command_scope_default),
                                                               attribute), 424
                                                      business_connection_id
BotCommandScopeType
                                  (class
                                                  in
                                                               (aiogram.methods.send animation.SendAnimation
        aiogram.enums.bot_command_scope_type),
                                                               attribute), 432
                                                      business_connection_id
BotDescription
                                                               (aiogram.methods.send_audio.SendAudio
                               (class
                                                  in
         aiogram.types.bot_description), 28
                                                               attribute), 435
BotName (class in aiogram.types.bot_name), 28
                                                      business_connection_id
                                                               (aiogram.methods.send\_chat\_action.SendChatAction
\verb|bots| (a iogram. dispatcher. middle wares. data. Middle ware Data
         attribute), 570
                                                               attribute), 438
BotShortDescription
                                  (class
                                                  in business_connection_id
         aiogram.types.bot_short_description), 28
                                                               (aiogram.methods.send_contact.SendContact
bottom_color(aiogram.types.background_fill_gradient.BackgroundEithGrade);n439
         attribute), 21
                                                      business_connection_id
                                                               (aiogram.methods.send dice.SendDice
BOWLING
           (aiogram.enums.dice_emoji.DiceEmoji
                                                 at-
                                                                                                        at-
         tribute), 544
                                                               tribute), 441
```

business_connection_id (aiogram.methods.send_document.SendDocument)	
attribute), 444	business_intro(aiogram.types.chat_full_info.ChatFullInfo
business_connection_id	attribute), 55
(aiogram.methods.send_game.SendGame	business_location (aiogram.types.chat.Chat at-
attribute), 517	tribute), 34
business_connection_id	business_location(aiogram.types.chat_full_info.ChatFullInfo attribute), 55
(aiogram.methods.send_location.SendLocation attribute), 447	BUSINESS_MESSAGE (aiogram.enums.update_type.UpdateType
business_connection_id	attribute), 552
	al Gusip ness_message (aiogram.types.update.Update at-
attribute), 450	tribute), 354
business_connection_id	business_opening_hours (aiogram.types.chat.Chat
(aiogram.methods.send_message.SendMessage	attribute), 34
attribute), 452	business_opening_hours
business_connection_id	(aiogram.types.chat_full_info.ChatFullInfo
(aiogram.methods.send_paid_media.SendPaidMe	
attribute), 455	BusinessConnection (class in
business_connection_id	aiogram.types.business_connection), 29
(aiogram.methods.send_photo.SendPhoto	BusinessIntro (class in aiogram.types.business_intro),
attribute), 457	29
business_connection_id	BusinessLocation (class in
(aiogram.methods.send_poll.SendPoll at-	aiogram.types.business_location), 30
tribute), 460	BusinessMessagesDeleted (class in
business_connection_id	$aiogram.types.business_messages_deleted),30$
$(a iogram.methods.send_sticker.SendSticker$	BusinessOpeningHours (class in
attribute), 368	aiogram.types.business_opening_hours),
business_connection_id	30
(aiogram.methods.send_venue.SendVenue	BusinessOpeningHoursInterval (class in
attribute), 463	aiogram.types.business_opening_hours_interval),
business_connection_id	31
(aiogram.methods.send_video.SendVideo	button(aiogram.methods.answer_inline_query.AnswerInlineQuery
attribute), 466	attribute), 511
business_connection_id	button_text(aiogram.types.web_app_data.WebAppData
(aiogram.methods.send_video_note.SendVideoNo	
<pre>attribute), 469 business_connection_id</pre>	buttons (aiogram.utils.keyboard.InlineKeyboardBuilder
(aiogram.methods.send_voice.SendVoice	property), 637 buttons (aiogram.utils.keyboard.ReplyKeyboardBuilder
attribute), 472	property), 638
business_connection_id	BYN (aiogram.enums.currency.Currency attribute), 542
(aiogram.methods.stop_message_live_location.St	· · · · · · · · · · · · · · · · · · ·
attribute), 507	C
business_connection_id	cache_time(aiogram.methods.answer_callback_query.AnswerCallbackQ
(aiogram.methods.stop_poll.StopPoll attribute),	attribute), 380
509	cache_time (aiogram.methods.answer_inline_query.AnswerInlineQuery
business_connection_id	attribute), 510
	act Marsageime (aiogram.utils.callback_answer.CallbackAnswer
attribute), 495	property), 651
business_connection_id	CAD (aiogram.enums.currency.Currency attribute), 542
	tesMessesesQetes(diogram.handlers.callback_query.CallbackQueryHandle
attribute), 30	property), 631
business_connection_id	callback_data(aiogram.types.inline_keyboard_button.InlineKeyboardB
(aiogram.types.message.Message attribute), 201	attribute), 181

```
callback_game (aiogram.types.inline_keyboard_button.InlineKeyboautthBbuttan), 426
                                           attribute), 182
                                                                                                                                                                                                                                                                 can_delete_stories(aiogram.types.chat_administrator_rights.ChatAdm
{\tt CALLBACK\_QUERY}\ (aiogram.enums.update\_type.UpdateType
                                                                                                                                                                                                                                                                                                            attribute), 49
                                           attribute), 552
                                                                                                                                                                                                                                                                 can_delete_stories(aiogram.types.chat_member_administrator.ChatM
callback_query (aiogram.types.update.Update
                                                                                                                                                                                                                                                                                                            attribute), 102
                                          tribute), 354
                                                                                                                                                                                                                                                                 can_edit_messages (aiogram.methods.promote_chat_member.PromoteClatering)
callback_query_id(aiogram.methods.answer_callback_query.AnswetnChllbackQuery
                                           attribute), 380
                                                                                                                                                                                                                                                                 can_edit_messages (aiogram.types.chat_administrator_rights.ChatAdmi
callback_query_without_state
                                                                                                                                                                                                                                                                                                            attribute), 49
                                           (aiogram.fsm.scene.SceneConfig
                                                                                                                                                                                                           attribute),
                                                                                                                                                                                                                                                                 can_edit_messages (aiogram.types.chat_member_administrator.ChatMe.
                                                                                                                                                                                                                                                                                                            attribute), 102
CallbackAnswer
                                                                                                                                                                                                                                                                 can_edit_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_
                                                                                                                                                 (class
                                          aiogram.utils.callback_answer), 651
                                                                                                                                                                                                                                                                                                            attribute), 426
CallbackAnswerException, 618
                                                                                                                                                                                                                                                                 can_edit_stories(aiogram.types.chat_administrator_rights.ChatAdmin
CallbackAnswerMiddleware
                                                                                                                                                                                                                                             in
                                                                                                                                                                                                                                                                                                            attribute), 49
                                                                                                                                                                              (class
                                          aiogram.utils.callback_answer), 650
                                                                                                                                                                                                                                                                 can_edit_stories (aiogram.types.chat_member_administrator.ChatMem
CallbackData (class in aiogram.filters.callback_data),
                                                                                                                                                                                                                                                                                                            attribute), 102
                                                                                                                                                                                                                                                                 can_invite_users(aiogram.methods.promote_chat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_member.PromoteChat_
CallbackGame (class in aiogram.types.callback_game),
                                                                                                                                                                                                                                                                                                             attribute), 426
                                                                                                                                                                                                                                                                 can_invite_users(aiogram.types.chat_administrator_rights.ChatAdmin
CallbackQuery
                                                                                                                                               (class
                                                                                                                                                                                                                                             in
                                                                                                                                                                                                                                                                                                            attribute), 49
                                          aiogram.types.callback_query), 31
                                                                                                                                                                                                                                                                 can_invite_users (aiogram.types.chat_member_administrator.ChatMem
CallbackQueryHandler
                                                                                                                                                                  (class
                                                                                                                                                                                                                                                                                                             attribute), 102
                                                                                                                                                                                                                                             in
                                          aiogram.handlers.callback_query), 631
                                                                                                                                                                                                                                                                 can_invite_users (aiogram.types.chat_member_restricted.ChatMember_
can_add_web_page_previews
                                                                                                                                                                                                                                                                                                            attribute), 105
                                          (aiogram.types.chat_member_restricted.ChatMembanRaxtwätael_users (aiogram.types.chat_permissions.ChatPermissions
                                          attribute), 105
                                                                                                                                                                                                                                                                                                            attribute), 128
can_add_web_page_previews
                                                                                                                                                                                                                                                                 can_join_groups (aiogram.types.user.User attribute),
                                          (aiogram.types.chat_permissions.ChatPermissions
                                          attribute), 128
                                                                                                                                                                                                                                                                 can\_manage\_chat (aiogram.methods.promote\_chat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_member.PromoteChat\_m
can_be_edited(aiogram.types.chat_member_administrator.ChatMentbabAub)uj#13frator
                                          attribute), 101
                                                                                                                                                                                                                                                                 \verb|can_manage_chat| (a iogram. types. chat\_administrator\_rights. ChatAdministrator\_rights. Chat
can_change_info(aiogram.methods.promote_chat_member.Promote@hibtMe)nb8r
                                                                                                                                                                                                                                                                 \verb|can_manage_chat| (a iogram. types. chat\_member\_administrator. Chat Member\_administrator. Chat Memb
                                          attribute), 426
can_change_info (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.ChatAdministra
                                          attribute), 49
                                                                                                                                                                                                                                                                 can_manage_topics (aiogram.methods.promote_chat_member.PromoteClass)
can_change_info(aiogram.types.chat_member_administrator.ChatMtembareA)dAllhistrator
                                          attribute), 102
                                                                                                                                                                                                                                                                 can_manage_topics (aiogram.types.chat_administrator_rights.ChatAdmi
can_change_info(aiogram.types.chat_member_restricted.ChatMembteribestericted)
                                          attribute), 105
                                                                                                                                                                                                                                                                 can_manage_topics (aiogram.types.chat_member_administrator.ChatMe.
can_change_info(aiogram.types.chat_permissions.ChatPermissionsattribute), 102
                                          attribute), 128
                                                                                                                                                                                                                                                                 can_manage_topics (aiogram.types.chat_member_restricted.ChatMembe
                                                                                                                                                                                                                                                                                                            attribute), 105
can_connect_to_business (aiogram.types.user.User
                                                                                                                                                                                                                                                                 can_manage_topics (aiogram.types.chat_permissions.ChatPermissions
                                          attribute), 271
                                                                                                                                                                                                                                                                                                            attribute), 128
can_delete_messages
                                          (aiogram.methods.promote_chat_member.Promotedahanaher_video_chats
                                                                                                                                                                                                                                                                                                            (aiogram.methods.promote_chat_member.PromoteChatMember
                                          attribute), 426
can_delete_messages
                                                                                                                                                                                                                                                                                                            attribute), 426
                                           (aiogram.types.chat_administrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatA
                                          attribute), 49
                                                                                                                                                                                                                                                                                                             (aiogram.types.chat_administrator_rights.ChatAdministratorRights)
can_delete_messages
                                                                                                                                                                                                                                                                                                             attribute), 49
                                          (aiogram.types.chat member administrator.ChatMambaadgenisideor_chats
                                          attribute), 101
                                                                                                                                                                                                                                                                                                             (aiogram.types.chat_member_administrator.ChatMemberAdminis
```

can_delete_stories(aiogram.methods.promote chat member.PromotedChei)Member

```
can_pin_messages(aiogram.methods.promote chat membenPsendteGlifttMainthum.types.chat full info.ChatFullInfo
                       attribute), 426
                                                                                                                                                                        attribute), 56
can_pin_messages(aiogram.types.chat administrator righanChenkl/messagesRighgsam.types.chat member restricted.ChatMembe
                                                                                                                                                                        attribute), 105
                       attribute), 49
can_pin_messages (aiogram.types.chat_member_administrators@hotMessbarges/hotiogratortypes.chat_permissions. ChatPermissions
                       attribute), 102
                                                                                                                                                                        attribute), 127
can_pin_messages (aiogram.types.chat member restricted ach a state of the control of the control
                                                                                                                                                                        (aiogram.types.chat member restricted.ChatMemberRestricted
                       attribute), 105
can_pin_messages (aiogram.types.chat permissions.ChatPermissionstribute), 105
                                                                                                                                               can_send_other_messages
                       attribute), 128
can_post_messages (aiogram.methods.promote_chat_member.PromoticighatMtsymbsechat_permissions.ChatPermissions
                       attribute), 426
                                                                                                                                                                        attribute), 128
can_post_messages (aiogram.types.chat_administrator_rights, & endAphaindistmediaRights
                                                                                                                                                                        (aiogram.types.chat_full_info.ChatFullInfo
                       attribute), 49
can_post_messages (aiogram.types.chat_member_administrator.ChatMbmdba)rAdministrator
                       attribute), 102
                                                                                                                                                can_send_photos(aiogram.types.chat_member_restricted.ChatMemberR
can_post_stories(aiogram.methods.promote_chat_member.Promote@thattel.grhb&r
                       attribute), 426
                                                                                                                                               can_send_photos (aiogram.types.chat_permissions.ChatPermissions
can_post_stories (aiogram.types.chat_administrator_rights.ChatAalmiibistedut&2Rights
                                                                                                                                                can_send_polls(aiogram.types.chat member restricted.ChatMemberRes
                       attribute), 49
can_post_stories (aiogram.types.chat_member_administrator.ChatMeihibter)AdMinistrator
                       attribute), 102
                                                                                                                                                can_send_polls(aiogram.types.chat permissions.ChatPermissions
can_promote_members
                                                                                                                                                                        attribute), 128
                       (aiogram.methods.promote chat member.Promote@hatshendbwideo_notes
                       attribute), 426
                                                                                                                                                                        (aiogram.types.chat member restricted.ChatMemberRestricted
can_promote_members
                                                                                                                                                                        attribute), 105
                        (aiogram.types.chat_administrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatA
                       attribute), 49
                                                                                                                                                                        (aiogram.types.chat_permissions.ChatPermissions
can_promote_members
                                                                                                                                                                        attribute), 128
                       (aiogram.types.chat_member_administrator.ChatManubseAdministeascaniogram.types.chat_member_restricted.ChatMemberR
                       attribute), 102
                                                                                                                                                                        attribute), 105
can_read_all_group_messages
                                                                                                                                                can_send_videos(aiogram.types.chat_permissions.ChatPermissions
                        (aiogram.types.user.User attribute), 271
                                                                                                                                                                        attribute), 128
can_reply(aiogram.types.business_connection.BusinessCommercsiemd_voice_notes
                       attribute), 29
                                                                                                                                                                        (aiogram.types.chat member restricted.ChatMemberRestricted
can_restrict_members
                                                                                                                                                                        attribute), 105
                       (aiogram.methods.promote chat member.Promote@hatshendbwoice_notes
                       attribute), 426
                                                                                                                                                                        (aiogram.types.chat_permissions.ChatPermissions
can_restrict_members
                                                                                                                                                                        attribute), 128
                       (aiogram.types.chat_administrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatAdministrator_rights.ChatA
                                                                                                                                                                                                                             (aiogram.types.chat.Chat
                       attribute), 49
                                                                                                                                                                        attribute), 34
can_restrict_members
                                                                                                                                               can_set_sticker_set
                        (aiogram.types.chat member administrator.ChatMemberAdvaingstratorpes.chat full info.ChatFullInfo
                       attribute), 102
                                                                                                                                                                        attribute), 57
can_send_after(aiogram.utils.web_app.WebAppInitDatacaption(aiogram.methods.copy_message.CopyMessage
                       attribute), 647
                                                                                                                                                                        attribute), 388
can_send_audios (aiogram.types.chat_member_restricted.CalputManhior@asmriatethods.edit_message_caption.EditMessageCaption
                       attribute), 105
                                                                                                                                                                        attribute), 499
can_send_audios (aiogram.types.chat_permissions.ChatPeaptsions(aiogram.methods.send_animation.SendAnimation
                       attribute), 127
                                                                                                                                                                        attribute), 433
can_send_documents(aiogram.types.chat_member_restrictad.tGlantMainbenRustricthadls.send_audio.SendAudio at-
                                                                                                                                                                        tribute), 435
                       attribute), 105
can_send_documents(aiogram.types.chat permissions.ClauBtionssiaingram.methods.send document.SendDocument
                       attribute), 128
                                                                                                                                                                        attribute), 444
```

```
caption(aiogram.methods.send_paid_media.SendPaidMedia
                                                                                                                                 attribute), 435
                  attribute), 455
                                                                                                               caption_entities(aiogram.methods.send document.SendDocument
caption (aiogram.methods.send photo.SendPhoto at-
                                                                                                                                 attribute), 445
                  tribute), 457
                                                                                                               caption_entities(aiogram.methods.send_paid_media.SendPaidMedia
caption
                              (aiogram.methods.send_video.SendVideo
                                                                                                                                 attribute), 455
                  attribute), 467
                                                                                                               caption_entities(aiogram.methods.send photo.SendPhoto
                                                                                                                                 attribute), 457
                              (aiogram.methods.send voice.SendVoice
caption
                                                                                                               {\tt caption\_entities} \ (a iogram. methods. send\_video. Send Video
                  attribute), 472
caption (aiogram.types.inline_query_result_audio.InlineQueryResultAtribate), 467
                  attribute), 283
                                                                                                               caption_entities (aiogram.methods.send_voice.SendVoice
caption(aiogram.types.inline_query_result_cached_audio.InlineQuartyRbsult)C4ThedAudio
                                                                                                               caption_entities(aiogram.types.inline_query_result_audio.InlineQuer
                  attribute), 285
caption (aiogram.types.inline_query_result_cached_document.Inline@nrebynResultCachedDocument
                                                                                                               caption_entities(aiogram.types.inline_query_result_cached_audio.Inli
                  attribute), 287
{\tt caption} \ (aiogram. types. in line\_query\_result\_cached\_gif. In line Query \textit{RetxnibtGe}); \textit{PostGif}
                  attribute), 289
                                                                                                               caption_entities(aiogram.types.inline_query_result_cached_document
caption (aiogram.types.inline_query_result_cached_mpeg4_gif.InlinaQuibuyAcadiiCachedMpeg4Gif
                  attribute), 291
                                                                                                               caption_entities (aiogram.types.inline_query_result_cached_gif.Inline
caption (aiogram.types.inline_query_result_cached_photo.InlineQuary#Nesult)Ca8hedPhoto
                  attribute), 294
                                                                                                               caption_entities(aiogram.types.inline_query_result_cached_mpeg4_g
caption (aiogram.types.inline_query_result_cached_video.InlineQueattRibsult)CachedVideo
                  attribute), 298
                                                                                                               caption_entities (aiogram.types.inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_photo.Inline_query_result_cached_ph
caption (aiogram.types.inline_query_result_cached_voice.InlineQuentiRidmuleCachedVoice
                                                                                                               caption_entities(aiogram.types.inline query result cached video.Inli
                  attribute), 300
caption (aiogram.types.inline_query_result_document.InlineQueryRastrifDoc)unDOst
                                                                                                              {\tt caption\_entities} \ (a iogram. types. in line\_query\_result\_cached\_voice. In line\_
                  attribute), 303
caption (aiogram.types.inline_query_result_gif.InlineQueryResultGifttribute), 300
                                                                                                               caption_entities(aiogram.types.inline_query_result_document.InlineQ
                  attribute), 305
caption (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryRetxiltMpgg4Gif
                  attribute), 310
                                                                                                               caption_entities(aiogram.types.inline_query_result_gif.InlineQueryRe
caption (aiogram.types.inline_query_result_photo.InlineQueryResultPhribate), 305
                  attribute), 312
                                                                                                               caption_entities (aiogram.types.inline_query_result_mpeg4_gif.Inline)
caption (aiogram.types.inline_query_result_video.InlineQueryResultMtdibute), 310
                                                                                                              caption_entities(aiogram.types.inline_query_result_photo.InlineQuery_
                  attribute), 316
caption (aiogram.types.inline_query_result_voice.InlineQueryResultMoitleute), 312
                  attribute), 317
                                                                                                              caption_entities(aiogram.types.inline_query_result_video.InlineQuery
caption (aiogram.types.input_media_animation.InputMediaAnimatiattribute), 316
                  attribute), 184
                                                                                                               caption_entities (aiogram.types.inline_query_result_voice.InlineQuery
{\tt caption}\ (aiogram.types.input\_media\_audio.InputMediaAudio
                                                                                                                                 attribute), 317
                                                                                                               caption_entities(aiogram.types.input_media_animation.InputMediaAr
                  attribute), 185
caption (aiogram.types.input media document.InputMediaDocumentttribute), 184
                                                                                                               {\tt caption\_entities} \ (a iogram. types. input\_media\_audio. InputMediaAudio
                  attribute), 186
{\tt caption} \ (a iogram. types. input\_media\_photo. Input Media Photo
                                                                                                                                 attribute), 185
                                                                                                               caption_entities(aiogram.types.input_media_document.InputMediaDa
                  attribute), 187
{\tt caption} (a iogram. types. input\_media\_video. Input Media Video
                                                                                                                                 attribute), 186
                  attribute), 188
                                                                                                               caption_entities(aiogram.types.input_media_photo.InputMediaPhoto
caption (aiogram.types.message.Message attribute), 203
                                                                                                                                 attribute), 187
caption_entities(aiogram.methods.copy_message.Copyddpstxign_entities(aiogram.types.input_media_video.InputMediaVideo
                  attribute), 388
                                                                                                                                 attribute), 188
caption_entities (aiogram.methods.edit_message_caption_tithde.sangicCinption (aiogram.types.message.Message
                  attribute), 499
                                                                                                                                 attribute), 203
caption_entities(aiogram.methods.send_animation.SenClASHIFIAGiomiogram.enums.message_entity_type.MessageEntityType
                  attribute), 433
                                                                                                                                 attribute), 548
caption_entities(aiogram.methods.send audio.SendAudiashTag(class in aiogram.utils.formatting), 656
```

```
certificate(aiogram.methods.set_webhook.SetWebhook
                                                                                                                                       attribute), 537
                   attribute), 534
                                                                                                                   CHAT_BACKGROUND_SET
                                                                                                                                       (aiogram.enums.content_type.ContentType
CHANNEL (aiogram.enums.chat_type.ChatType attribute),
                                                                                                                                       attribute), 541
CHANNEL (aiogram.enums.message_origin_type.MessageOrighiallypeackground_set
                  attribute), 549
                                                                                                                                       (aiogram.types.message.Message
                                                                                                                                                                                                               attribute),
CHANNEL_CHAT_CREATED
                                                                                                                    CHAT_BOOST (aiogram.enums.update_type.UpdateType
                   (aiogram.enums.content_type.ContentType
                   attribute), 540
                                                                                                                                       attribute), 553
channel_chat_created
                                                                                                                    chat_boost (aiogram.types.update.Update attribute),
                   (aiogram.types.message.Message
                                                                                           attribute),
                                                                                                                    chat_has_username(aiogram.types.keyboard_button_request_chat.Keybo
CHANNEL_POST (aiogram.enums.update_type.UpdateType
                                                                                                                                       attribute), 193
                  attribute), 552
                                                                                                                    chat_id (aiogram.methods.approve_chat_join_request.ApproveChatJoinRe
channel_post (aiogram.types.update.Update attribute),
                                                                                                                                       attribute), 381
                                                                                                                    \verb|chat_id| (a iogram.methods.ban\_chat\_member.BanChatMember|
CHAT (aiogram.enums.bot_command_scope_type.BotCommandScopeTatpeibute), 382
                                                                                                                    \verb|chat_id| (a iogram.methods.ban_chat\_sender\_chat.BanChatSenderChat|
                   attribute), 537
CHAT (aiogram.enums.message_origin_type.MessageOriginType
                                                                                                                                       attribute), 383
                                                                                                                    chat_id(aiogram.methods.close_forum_topic.CloseForumTopic
                   attribute), 549
CHAT (aiogram.enums.transaction_partner_type.TransactionPartnerTypteribute), 385
                   attribute), 552
                                                                                                                    {\tt chat\_id}\ (aiogram.methods.close\_general\_forum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_topic.CloseGeneralForum\_to
CHAT (aiogram.fsm.strategy.FSMStrategy attribute), 595
                                                                                                                                       attribute), 386
chat (aiogram.types.business messages deleted.BusinessMeksatzeisDeletatzeram.methods.copy message.CopyMessage
                   attribute), 30
                                                                                                                                       attribute), 388
chat (aiogram.types.chat_boost_removed.ChatBoostRemovehat_id (aiogram.methods.copy_messages.CopyMessages
                   attribute), 51
                                                                                                                                       attribute), 389
\verb|chat| (aiogram.types.chat\_boost\_updated.ChatBoostUpdated.hat\_id (aiogram.methods.create\_chat\_invite\_link.CreateChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteLink.createChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInviteChatInvi
                                                                                                                                       attribute), 391
                   attribute), 53
chat (aiogram.types.chat_join_request.ChatJoinRequest chat_id (aiogram.methods.create_chat_subscription_invite_link.CreateCh
                  attribute), 58
                                                                                                                                       attribute), 392
chat (aiogram.types.chat_member_updated.ChatMemberUpdhatedid (aiogram.methods.create_forum_topic.CreateForumTopic
                   attribute), 106
                                                                                                                                       attribute), 394
chat (aiogram.types.external_reply_info.ExternalReplyInfochat_id (aiogram.methods.decline_chat_join_request.DeclineChatJoinRed
                   attribute), 132
                                                                                                                                       attribute), 395
chat (aiogram.types.giveaway_winners.GiveawayWinners chat_id (aiogram.methods.delete_chat_photo.DeleteChatPhoto
                   attribute), 137
                                                                                                                                       attribute), 396
chat (aiogram.types.inaccessible_message.InaccessibleMessheet_id (aiogram.methods.delete_chat_sticker_set.DeleteChatStickerSet
                   attribute), 138
                                                                                                                                       attribute), 397
chat (aiogram.types.message.Message attribute), 201
                                                                                                                   chat_id(aiogram.methods.delete_forum_topic.DeleteForumTopic
chat (aiogram.types.message_origin_channel.MessageOriginChannelattribute), 398
                   attribute), 257
                                                                                                                    chat_id(aiogram.methods.delete_message.DeleteMessage
chat (aiogram.types.message_reaction_count_updated.MessageReaction(buta)LADdated
                   attribute), 259
                                                                                                                    \verb|chat_id| (a iogram.methods.delete\_messages.DeleteMessages|
chat (aiogram.types.message_reaction_updated.MessageReactionUpdateibute), 498
                                                                                                                    chat_id (aiogram.methods.edit_chat_invite_link.EditChatInviteLink
                   attribute), 259
chat (aiogram.types.story.Story attribute), 269
                                                                                                                                       attribute), 400
chat (aiogram.types.transaction_partner_chat.TransactionRharterIChatiogram.methods.edit_chat_subscription_invite_link.EditChatSu
                   attribute), 336
                                                                                                                                       attribute), 402
chat (aiogram.utils.web_app.WebAppInitData attribute), chat_id (aiogram.methods.edit_forum_topic.EditForumTopic
                                                                                                                                       attribute), 403
Chat (class in aiogram.types.chat), 32
                                                                                                                    chat_id(aiogram.methods.edit_general_forum_topic.EditGeneralForumTe
CHAT_ADMINISTRATORS
                                                                                                                                       attribute), 404
```

- attribute), 499 attribute), 439
- chat_id(aiogram.methods.edit_message_live_location.EditMaxsaigaLivediogationmethods.send_dice.SendDice atattribute), 501 tribute), 441
- chat_id (aiogram.methods.edit_message_media.EditMessaghMedid (aiogram.methods.send_document.SendDocument attribute), 503 attribute), 444
- chat_id(aiogram.methods.edit_message_reply_markup.EditMatssingeReplyMaiorgupm.methods.send_game.SendGame attribute), 504
- chat_id (aiogram.methods.export_chat_invite_link.Export**Chart**[nide(laioo]gram.methods.send_invoice.SendInvoice atattribute), 405 tribute), 528
- chat_id(aiogram.methods.forward_message.ForwardMessaget_id (aiogram.methods.send_location.SendLocation attribute), 406 attribute), 447
- chat_id (aiogram.methods.forward_messages.ForwardMesshgds_id (aiogram.methods.send_media_group.SendMediaGroup attribute), 407
 attribute), 449
- chat_id (aiogram.methods.get_chat.GetChat attribute), chat_id (aiogram.methods.send_message.SendMessage 409 attribute), 452
- chat_id (aiogram.methods.get_chat_administrators.GetChathathninds(ninogram.methods.send_paid_media.SendPaidMedia attribute), 410 attribute), 454
- chat_id(aiogram.methods.get_chat_member.GetChatMembbat_id (aiogram.methods.send_photo.SendPhoto atattribute), 411 tribute), 457
- chat_id(aiogram.methods.get_chat_member_count.GetCh**ahldx**m**ba**rCo(**ai**ogram.methods.send_poll.SendPoll at-attribute), 412 tribute), 460
- chat_id(aiogram.methods.get_chat_menu_button.GetChat**\text{Matu_But**toniogram.methods.send_sticker.SendSticker atattribute), 412 tribute), 368
- chat_id(aiogram.methods.get_game_high_scores.GetGameHagh_Scoresiogram.methods.send_venue.SendVenue atattribute), 516 tribute), 463
- chat_id(aiogram.methods.get_user_chat_boosts.GetUser@hatBoists (aiogram.methods.send_video.SendVideo attribute), 419 attribute), 466
- chat_id(aiogram.methods.hide_general_forum_topic.Hide**Chate_ràtF**(aiogiRapimethods.send_video_note.SendVideoNote attribute), 421 attribute), 469
- chat_id (aiogram.methods.leave_chat.LeaveChat chat_id (aiogram.methods.send_voice.SendVoice attribute), 422 attribute), 472
- chat_id (aiogram.methods.pin_chat_message.PinChatMessahget_id (aiogram.methods.set_chat_administrator_custom_title.SetChatAcattribute), 424

 attribute), 474
- chat_id (aiogram.methods.promote_chat_member.PromoteChatMedr(baiogram.methods.set_chat_description.SetChatDescription attribute), 425 attribute), 475
- chat_id (aiogram.methods.remove_chat_verification.Remove\attat\deltat\de
- chat_id (aiogram.methods.reopen_forum_topic.ReopenForumfopid (aiogram.methods.set_chat_permissions.SetChatPermissions attribute), 427 attribute), 427
- chat_id (aiogram.methods.reopen_general_forum_topic.Rechpatr_Geducut Forumilephiads.set_chat_photo.SetChatPhoto attribute), 428 attribute), 478
- chat_id(aiogram.methods.restrict_chat_member.RestrictChheMeindeniogram.methods.set_chat_sticker_set.SetChatStickerSet attribute), 429 attribute), 479
- chat_id(aiogram.methods.revoke_chat_invite_link.Revoke**ChatI**riv**it**eLiidiogram.methods.set_chat_title.SetChatTitle attribute), 430 attribute), 480
- chat_id (aiogram.methods.send_animation.SendAnimationchat_id (aiogram.methods.set_game_score.SetGameScore attribute), 432 attribute), 519
- chat_id(aiogram.methods.send_chat_action.SendChatActi**oh**at_id(aiogram.methods.stop_message_live_location.StopMessageLiveLattribute), 438 attribute), 508
- chat_id (aiogram.methods.send contact.SendContact chat_id (aiogram.methods.stop poll.StopPoll attribute),

509	tribute), 278	
chat_id(aiogram.methods.unban_chat_member.UnbanChattribute), 489		at-
chat_id (aiogram.methods.unban_chat_sender_chat.Unba		8
attribute), 490		in
chat_id(aiogram.methods.unhide_general_forum_topic.U		
attribute), 491	ChatActionSender (class in aiogram.utils.chat_action	n).
chat_id(aiogram.methods.unpin_all_chat_messages.Unp		-//
attribute), 492	ChatAdministratorRights (class	in
chat_id(aiogram.methods.unpin_all_forum_topic_messagattribute), 493		
chat_id(aiogram.methods.unpin_all_general_forum_topi	cChart Backschrouincall General Forum Alousic Messages	in
attribute), 494	aiogram.types.chat_background), 50	
<pre>chat_id(aiogram.methods.unpin_chat_message.UnpinChat_message.</pre>		
attribute), 495		in
chat_id (aiogram.methods.verify_chat.VerifyChat	aiogram.types.chat_boost_added), 50	
attribute), 378	ChatBoostRemoved (class	in
<pre>chat_id(aiogram.types.bot_command_scope_chat.BotCom</pre>	nmandScopic@naum.types.chat_boost_removed), 51	
attribute), 26		in
<pre>chat_id(aiogram.types.bot_command_scope_chat_admin</pre>	istrators.B atiGgramatydSs.apetC_lbatAst <u>m</u> si aistre t)o 5 s	
attribute), 27	ChatBoostSourceGiftCode (class	in
<pre>chat_id(aiogram.types.bot_command_scope_chat_memb</pre>	er.BotCom ıniaydSıvap,q&k.adMienbloa st_source_gift_code) 51	,
<pre>chat_id (aiogram.types.chat_shared.ChatShared at-</pre>	ChatBoostSourceGiveaway (class	in
tribute), 129	aiogram.types.chat_boost_source_giveaway)	,
$\verb chat_id (a iogram. types. reply_parameters. ReplyParameters) $	s 52	
attribute), 267	ChatBoostSourcePremium (class	in
<pre>chat_instance(aiogram.types.callback_query.CallbackQ</pre>	Query aiogram.types.chat_boost_source_premium), 52	
$\verb chat_instance (a iogram.utils.web_app.WebAppInitData $	ChatBoostSourceType (class	in
attribute), 647	<pre>aiogram.enums.chat_boost_source_type),</pre>	
$\verb chat_is_channe (a iogram. types. keyboard_button_requently and all the properties of the propert$	st_chat.Ke\$B&ardButtonRequestChat	
attribute), 193	ChatBoostUpdated (class	in
$\verb chat_is_created (a iogram. types. keyboard_button_requerd) $	st_chat.KeydbogmahButstpoesRelqaute_sh&bsstt_updated), 53	
attribute), 193	ChatFullInfo (class in aiogram.types.chat_full_info	o),
$\verb chat_is_forum (a iogram. types. keyboard_button_request_butto$		
attribute), 193	ChatInviteLink (class	in
${\tt CHAT_JOIN_REQUEST}\ (a iogram.enums.update_type.U$		
attribute), 553	ChatJoinRequest (class	in
chat_join_request (aiogram.types.update.Update at-	aiogram.types.chat_join_request), 58	
tribute), 354	ChatLocation (class in aiogram.types.chat_location	n),
CHAT_MEMBER (aiogram.enums.bot_command_scope_type		
attribute), 537	ChatMember (class in aiogram.types.chat_member), 10	
CHAT_MEMBER (aiogram.enums.update_type.UpdateType	ChatMemberAdministrator (class	in
attribute), 553	aiogram.types.chat_member_administrator),	
chat_member (aiogram.types.update.Update attribute),	101	
354	ChatMemberBanned (class	in
CHAT_SHARED (aiogram.enums.content_type.ContentType	aiogram.types.chat_member_banned), 102	
attribute), 540	ChatMemberLeft (class	in
chat_shared (aiogram.types.message.Message at-	aiogram.types.chat_member_left), 103	;
tribute), 204 CHAT_TOPIC (aiogram.fsm.strategy.FSMStrategy at-	ChatMemberMember (class aiogram.types.chat_member_member), 103	in
CHAT_TOPIC (aiogram.fsm.strategy.FSMStrategy attribute), 595		in
chat_type (aiogram.types.inline_query.InlineQuery at-	aiogram.types.chat_member_owner), 104	ııı
chac_cype (anogram.types.mane_query.maneQuery at-	aiogram.iypes.chai_member_owner), 104	

ChatMemberRestricted	(class	in	CloseGeneralForumTopic (class in
aiogram.types.chat_n 104	nember_restricted),		aiogram.methods.close_general_forum_topic), 386
ChatMemberStatus	(class	in	${\tt closing_minute} \ (a iogram. types. business_opening_hours_interval. Business_opening_hours_in$
aiogram.enums.chat_	_member_status), 539		attribute), 31
ChatMemberUpdated	(class	in	CLP (aiogram.enums.currency.Currency attribute), 542
	nember_updated), 106		CNY (aiogram.enums.currency.Currency attribute), 542
ChatMemberUpdatedFilter	(class member_updated), 573	in	CODE (aiogram.enums.message_entity_type.MessageEntityType attribute), 548
ChatPermissions	(class	in	Code (class in aiogram.utils.formatting), 657
aiogram.types.chat_p	*		color (aiogram.types.background_fill_solid.BackgroundFillSolid
ChatPhoto (class in aiogram.t			attribute), 22
chats (aiogram.types.giveawa		136	colors (aiogram.types.background_fill_freeform_gradient.BackgroundFill_
ChatShared (class in aiogram			attribute), 21
ChatType (class in aiogram.en			command (aiogram.filters.command.CommandObject at-
check_flags() (in module		gs).	tribute), 572
620	and grammanspareners, and	5~/,	command (aiogram.types.bot_command.BotCommand at-
check_response() (aiogram.	.client.session.base.Bas	eSess	
<i>method</i>), 15			Command (class in aiogram.filters.command), 571
<pre>check_webapp_signature()</pre>) (in mod	lule	CommandObject (class in aiogram.filters.command), 572
aiogram.utils.web_ap			COMMANDS (aiogram.enums.menu_button_type.MenuButtonType
CHF (aiogram.enums.currency.		2	attribute), 547
			ocommands (aiogram.methods.set_my_commands.SetMyCommands
attribute), 547			attribute), 483
CHOOSE_STICKER (aiogram.en	ums.chat action.ChatA	ction	
attribute), 538			(aiogram.types.affiliate_info.AffiliateInfo
choose_sticker() (aiogram.	utils.chat action.Chat	Action	
class method), 644			commission_per_mille
CHOSEN_INLINE_RESULT			(aiogram.types.transaction_partner_affiliate_program.Transactio
(aiogram.enums.upda	ate_tvpe.UpdateTvpe		attribute), 336
attribute), 552	= 7F · · · · F · · · · · 7F ·		CONNECTED_WEBSITE (aiogram.enums.content_type.ContentType
chosen_inline_result (aio	gram.types.update.Upd	late	attribute), 540
attribute), 354			connected_website (aiogram.types.message.Message
ChosenInlineResult	(class	in	attribute), 204
aiogram.types.choser	•		ConstI18nMiddleware (class in
city (aiogram.types.shipping		ress	aiogram.utils.i18n.middleware), 641
attribute), 331	_ 11 0		CONTACT (aiogram.enums.content_type.ContentType at-
clear_data() (aiogra	am.fsm.scene.SceneWiz	ard	tribute), 540
method), 612	·		CONTACT (aiogram.enums.inline_query_result_type.InlineQueryResultType
ClientDecodeError, 619			attribute), 546
Close (class in aiogram.metho	ods.close), 384		<pre>contact (aiogram.types.external_reply_info.ExternalReplyInfo</pre>
close() (aiogram.client	t.session.base.BaseSess	sion	attribute), 132
method), 15			contact (aiogram.types.message.Message attribute), 203
close() (aiogram.fsm.scene	ScenesManager metho	(d),	Contact (class in aiogram.types.contact), 129
610			content_type (aiogram.types.message.Message prop-
close() (aiogram.fsm	n.storage.base.BaseStor	age	erty), 206
method), 595			ContentType (class in aiogram.enums.content_type),
<pre>close() (aiogram.webhook.ai</pre>	ohttp_server.SimpleRec	questl	Handler 539
method), 622			COP (aiogram.enums.currency.Currency attribute), 542
close_date (aiogram.method	ds.send_poll.SendPoll	at-	copy() (aiogram.utils.keyboard.InlineKeyboardBuilder
tribute), 461			method), 637
<pre>close_date (aiogram.types.pe</pre>	oll.Poll attribute), 263		copy() (aiogram.utils.keyboard.ReplyKeyboardBuilder
CloseForumTopic	(class	in	method), 638
aiogram.methods.clo	se forum topic), 385		<pre>copy_text(aiogram.types.inline keyboard button.InlineKeyboardButton</pre>

attribute), 182	attribute), 57
copy_to() (aiogram.types.message.Message method),	CREATOR (aiogram.enums.chat_member_status.ChatMemberStatus
245	attribute), 539
CopyMessage (class in aiogram.methods.copy_message),	creator (aiogram.types.chat_invite_link.ChatInviteLink
387	attribute), 57
	credentials (aiogram.types.passport_data.PassportData
aiogram.methods.copy_messages), 389	attribute), 345
	currency (aiogram.methods.create_invoice_link.CreateInvoiceLink
aiogram.types.copy_text_button), 130	attribute), 523
correct_option_id(aiogram.methods.send_poll.SendPo	
attribute), 461	attribute), 529
correct_option_id (aiogram.types.poll.Poll attribute),	
263	
	attribute), 321
country_code (aiogram.types.shipping_address.Shipping_attribute) 221	
attribute), 331	currency (aiogram.types.pre_checkout_query.PreCheckoutQuery
country_codes (aiogram.types.giveaway.Giveaway at-	attribute), 327
tribute), 136	currency (aiogram.types.refunded_payment.RefundedPayment
cover (aiogram.methods.send_video.SendVideo at-	attribute), 328
tribute), 467	currency (aiogram.types.successful_payment.SuccessfulPayment
cover (aiogram.types.input_media_video.InputMediaVideo	
attribute), 188	Currency (class in aiogram.enums.currency), 541
	Medisal bine description (aiogram.methods.verify_chat.VerifyChat
attribute), 190	attribute), 378
cover (aiogram.types.video.Video attribute), 274	custom_description(aiogram.methods.verify_user.VerifyUser
CRC (aiogram.enums.currency.Currency attribute), 542	attribute), 379
create_invite_link() (aiogram.types.chat.Chat	CUSTOM_EMOJI (aiogram.enums.message_entity_type.MessageEntityType
method), 39	attribute), 548
	CUSTOM_EMOJI (aiogram.enums.reaction_type_type.ReactionTypeType
aiogram.utils.deep_linking), 663	attribute), 550
create_startapp_link() (in module	
aiogram.utils.deep_linking), 664	attribute), 551
	<pre>custom_emoji_id(aiogram.methods.set_custom_emoji_sticker_set_thun</pre>
aiogram.utils.deep_linking), 663	attribute), 370
CreateChatInviteLink (class in	custom_emoji_id(aiogram.types.message_entity.MessageEntity
aiogram.methods.create_chat_invite_link),	attribute), 256
391	$\verb custom_emoji_id (a iogram. types. reaction_type_custom_emoji. Reaction_emoji. Reaction_emoji.$
CreateChatSubscriptionInviteLink (class in	attribute), 265
aiogram.methods.create_chat_subscription_invit	e <u>c</u> lisk)pm_emoji_id (aiogram.types.sticker.Sticker at-
392	tribute), 342
	$\verb custom_emoji_ids (a iogram.methods.get_custom_emoji_stickers.GetC$
<pre>aiogram.methods.create_forum_topic), 393</pre>	attribute), 362
CreateInvoiceLink (class in	custom_emoji_sticker_set_name
aiogram.methods.create_invoice_link), 523	(aiogram.types.chat.Chat attribute), 34
CreateNewStickerSet (class in	custom_emoji_sticker_set_name
aiogram.methods.create_new_sticker_set),	(aiogram.types.chat_full_info.ChatFullInfo
358	attribute), 57
creates_join_request	$\verb custom_title (a iogram.methods.set_chat_administrator_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_custom_title.Set_$
(aiogram.methods.create_chat_invite_link.Create	eChatInvite liink bute), 474
attribute), 391	$\verb"custom_title" (a iogram. types. chat_member_administrator. ChatMember_administrator. ChatMem$
creates_join_request	attribute), 102
(aiogram.methods.edit_chat_invite_link.EditChat	tlandstbinktitle(aiogram.types.chat_member_owner.ChatMemberOwner
attribute), 400	attribute), 104
creates_join_request	
ereaces_join_request	CustomEmoji (class in aiogram.utils.formatting), 658

D				aiogram.utils.deep_li	inking), 664	
	_theme_dimming(aiogram.types.backgroun attribute), 23			annouse), 557		
dark	_theme_dimming (aiogram.types.backgroun attribute), 24	d_type_v	valtpaper.	Haingramanypewall attribute), 547	u_button_type.Me	nuButtonType
	(aiogram.enums.dice_emoji.DiceEmoji att 544		Default	:BotProperties aiogram.client.defau	(class lt), 557	in
DART	(aiogram.types.dice.DiceEmoji attribute), 13	30		KeyBuilder	(class	in
DATA	(aiogram.enums.passport_element_error_ty, attribute), 549	pe.Passpo	ortElemen delete() (aiogram.iypes.me	base), 593 essage.Message	method),
data	(aiogram.types.callback_query.CallbackQu tribute), 31	ery at-	DELETE_	251 _CHAT_PHOTO(aiogran	n.enums.content_t	ype.ContentType
	(aiogram.types.encrypted_credentials.Encry attribute), 344	-	delete_	attribute), 540 _chat_photo (aiogra	am.types.message.	Message
data	(aiogram.types.encrypted_passport_element attribute), 345	.Encrypt	edPasspor delete_	_irom_set() (an	ogram.types.sticke	er.Sticker
data	(aiogram.types.passport_data.PassportDatribute), 345	ta at-	delete_	method),343 _message() (aiogram	a.types.chat.Chat	method),
data	(aiogram.types.web_app_data.WebAppDatribute), 277			38 _photo() (aiogram.ty _l		hod), 47
data	_hash(aiogram.types.passport_element_errattribute), 347	or_data_	fi eld.P as _{sj}	(aiogram.types.messa	ıField age.Message	method),
date	(aiogram.types.business_connection.Busines attribute), 29	ssConnec	tion delete_	249 sticker_set()	(aiogram.types.c	hat.Chat
date	(aiogram.types.chat_join_request.ChatJoin.attribute), 58	Request	DeleteC	method), 41 ChatPhoto	(class	in
date	(aiogram.types.chat_member_updated.Chata attribute), 106	MemberU	<i>Jpdated</i> DeleteO	aiogram.methods.del ChatStickerSet	(class	in
date	(aiogram.types.inaccessible_message.Inacce attribute), 138	essibleMe	_	aiogram.methods.del 397		set),
date	(aiogram.types.message.Message attribute),	201		_BUSINESS_MESSAGE		
	(aiogram.types.message_origin_channel.Me attribute), 257			attribute), 552		vpe
date	(aiogram.types.message_origin_chat.Messagattribute), 257	geOrigin	<i>C</i> haleted	l_business_message (aiogram.types.updat	es te.Update a	ttribute),
date	(aiogram.types.message_origin_hidden_use attribute), 258	r.Messago	<i>eOriginHi</i> DeleteF	TorumTopic	(class	in
	(aiogram.types.message_origin_user.Messagattribute), 258		DeleteM		(class	in
	(aiogram.types.message_reaction_count_upo attribute), 259		Detere	iessages	(ciass	in
	(aiogram.types.message_reaction_updated.Mattribute), 260		Detere	ryCommanus	(ciass	ιn
date	(aiogram.types.revenue_withdrawal_state_statte_stattibute), 330	ucceeded		399		
date	(aiogram.types.star_transaction.StarTransactionbute), 333	ction at-	DeleteS	StickerFromSet aiogram.methods.del	(class ete_sticker_from_	in set),
day ($aiogram.types.birthdate.Birthdate\ attribute),$	24	Dolo+o2	359	(alana	in
	ine() (aiogram.types.chat_join_request.Cha method), 59	ıtJoinReq		atogram.metnoas.aet		
Decl	ineChatJoinRequest (class	in	DeleteW	lebhook aiogram.methods.del	(class	in 1
	aiogram.methods.decline_chat_join_req 395	uest),	descrir			link.CreateInvoiceLink
deco		module	1	attribute), 523		-

${\tt description} ({\it aiogram.methods.send_invoice.SendInvoice}$		attribute), 445
attribute), 529		_content_type_detection
		(timgram.types.input_media_document.InputMediaDocument
attribute), 475		attribute), 186
description(aiogram.methods.set_my_description.SetMy		
attribute), 485		(aiogram.methods.set_game_score.SetGameScore
description(aiogram.types.bot_command.BotCommand		attribute), 519
attribute), 25		_notification
description (aiogram.types.bot_description.BotDescripti attribute), 28		(aiogram.client.default.DefaultBotProperties attribute), 557
		_notification
description (aiogram.types.chat_full_info.ChatFullInfo		(aiogram.methods.copy_message.CopyMessage
attribute), 56		attribute), 388
description (aiogram.types.game.Game attribute), 356		
description (aiogram.types.inline_query_result_article.In		
attribute), 281		attribute), 390
description(aiogram.types.inline_query_result_cached_		
attribute), 287		(aiogram.methods.forward_message.ForwardMessage
description(aiogram.types.inline_query_result_cached_		
		notification
description(aiogram.types.inline_query_result_cached_	video.Inlir	naQonemyResneHGalshfeatWiahab_messages.ForwardMessages
attribute), 298		attribute), 407
description(aiogram.types.inline_query_result_document	nd.Instatod@i	unoxHesiulta Daiconnent
attribute), 303		(aiogram.methods.pin_chat_message.PinChatMessage
${\tt description} ({\it aiogram.types.inline_query_result_photo.In}$		
		notification
<pre>description(aiogram.types.inline_query_result_video.In</pre>		Raintelviidenethods.send_animation.SendAnimation attribute), 433
description(aiogram.types.input_invoice_message_conte		
attribute), 321	_	(aiogram.methods.send_audio.SendAudio
description (aiogram.types.invoice.Invoice attribute),		attribute), 436
326	disable_	_notification
deserialize_telegram_object() (in module		(aiogram.methods.send_contact.SendContact
aiogram.utils.serialization), 665		attribute), 440
		_notification
module aiogram.utils.serialization), 665		(aiogram.methods.send_dice.SendDice at-
DeserializedTelegramObject (class in		tribute), 442
aiogram.utils.serialization), 665		notification
DetailedAiogramError, 618		(aiogram.methods.send_document.SendDocument
DICE (aiogram.enums.content_type.ContentType at-		attribute), 445
		notification
DICE (aiogram.enums.dice_emoji.DiceEmoji attribute), 544		(aiogram.methods.send_game.SendGame attribute), 517
DICE (aiogram.types.dice.DiceEmoji attribute), 130		notification
$\verb+dice+ (a iogram. types. external_reply_info. External ReplyInfo$		(aiogram.methods.send_invoice.SendInvoice
attribute), 132		attribute), 530
		notification
Dice (class in aiogram.types.dice), 130		(aiogram.methods.send_location.SendLocation
DiceEmoji (class in aiogram.enums.dice_emoji), 544		attribute), 448
DiceEmoji (class in aiogram.types.dice), 130		_notification
<pre>disable() (aiogram.utils.callback_answer.CallbackAnswe</pre>		(aiogram.methods.send_media_group.SendMediaGroup attribute), 450
disable_content_type_detection		notification
(aiogram.methods.send_document.SendDocumen	t	(aiogram.methods.send_message.SendMessage

<pre>attribute), 452 disable_notification</pre>	<pre>document (aiogram.types.background_type_wallpaper.BackgroundTypeWa attribute), 24</pre>
	ediacument (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 132
disable_notification (aiogram.methods.send_photo.SendPhoto	document (aiogram.types.message.Message attribute), 202
attribute), 458	Document (class in aiogram.types.document), 130
disable_notification (aiogram.methods.send_poll.SendPoll at-	<pre>document_file_id(aiogram.types.inline_query_result_cached_document</pre>
tribute), 461 disable_notification	document_url (aiogram.types.inline_query_result_document.InlineQuery_attribute), 302
(aiogram.methods.send_sticker.SendSticker attribute), 368	DOP (aiogram.enums.currency.Currency attribute), 542 download() (aiogram.client.bot.Bot method), 554
disable_notification	<pre>download_file() (aiogram.client.bot.Bot method), 553</pre>
(aiogram.methods.send_venue.SendVenue attribute), 464	DRIVER_LICENSE (aiogram.enums.encrypted_passport_element.Encrypted attribute), 545
disable_notification	drop_pending_updates
(aiogram.methods.send_video.SendVideo attribute), 467	(aiogram.methods.delete_webhook.DeleteWebhook attribute), 531
disable_notification	drop_pending_updates
(aiogram.methods.send_video_note.SendVideoNo attribute), 470	attribute), 535
disable_notification (aiogram.methods.send_voice.SendVoice	duration (aiogram.methods.send_animation.SendAnimation attribute), 433
attribute), 472	duration (aiogram.methods.send_audio.SendAudio at-
disable_web_page_preview	tribute), 436
(aiogram.methods.edit_message_text.EditMessag attribute), 506	ge ffart ation (aiogram.methods.send_video.SendVideo at- tribute), 466
disable_web_page_preview	<pre>duration(aiogram.methods.send_video_note.SendVideoNote</pre>
(aiogram.methods.send_message.SendMessage attribute), 453	duration (aiogram.methods.send_voice.SendVoice at-
disable_web_page_preview	tribute), 472
attribute), 324	t TexrMeiso ge C(mieg tam.types.animation.Animation at- tribute), 19
disabled(aiogram.utils.callback_answer.CallbackAnswer	
property), 651 dispatcher (aiogram.dispatcher.middlewares.data.Middlewares	duration(aiogram.types.input_media_animation.InputMediaAnimation ewareData attribute), 184
attribute), 570 Dispatcher (class in aiogram.dispatcher.dispatcher),	duration(aiogram.types.input_media_audio.InputMediaAudio attribute), 185
565	${\tt duration} ({\it aiogram.types.input_media_video.InputMediaVideo}$
distance (aiogram.types.proximity_alert_triggered.Proxin	
attribute), 264 DKK (aiogram.enums.currency.Currency attribute), 542	duration(aiogram.types.input_paid_media_video.InputPaidMediaVideo attribute), 190
do() (aiogram.types.chat.Chat method), 40	duration(aiogram.types.paid_media_preview.PaidMediaPreview
DOCUMENT (aiogram.enums.content_type.ContentType attribute), 539	attribute), 261 duration (aiogram.types.video.Video attribute), 274
	QuenayReonl(Tippgram.types.video_chat_ended.VideoChatEnded
attribute), 545	attribute), 275
DOCUMENT (aiogram.enums.input_media_type.InputMediaTattribute), 546	tribute), 276
<pre>document (aiogram.methods.send_document.SendDocume</pre>	nduration (aiogram.types.voice.Voice attribute), 276 DZD (aiogram.enums.currency.Currency attribute), 542
document (aiogram.types.background_type_pattern.Backg attribute), 23	

E	EditMessageText (class in
edit_caption() (aiogram.types.message.Message	aiogram.methods.edit_message_text), 505
method), 250	EditUserStarSubscription (class in
edit_date (aiogram.types.message.Message attribute), 202	aiogram.methods.edit_user_star_subscription), 525
edit_invite_link() (aiogram.types.chat.Chat method), 39	effect_id (aiogram.types.message.Message attribute), 202
memoa), 39 edit_live_location()	EGP (aiogram.enums.currency.Currency attribute), 542
(aiogram.types.message.Message method), 249	<pre>element_hash(aiogram.types.passport_element_error_unspecified.Passport_element_element_element_element_element_element_element_element_element_element_element_element_element_element_ele</pre>
edit_media() (aiogram.types.message.Message method), 248	EMAIL (aiogram.enums.encrypted_passport_element.EncryptedPassportEleattribute), 545
edit_reply_markup()	EMAIL (aiogram.enums.message_entity_type.MessageEntityType
(aiogram.types.message.Message method),	attribute), 548
248	email(aiogram.types.encrypted_passport_element.EncryptedPassportElem
<pre>edit_text() (aiogram.types.message.Message method),</pre>	attribute), 345
247	email (aiogram.types.order_info.OrderInfo attribute), 327
EditChatInviteLink (class in	Email (class in aiogram.utils.formatting), 656
aiogram.methods.edit_chat_invite_link),	EMOJI (aiogram.enums.reaction_type_type.ReactionTypeType
400 EditChatSubscriptionInviteLink (class in	attribute), 550
aiogram methods edit chat subscription invite	limp,ji (aiogram.methods.send_dice.SendDice attribute),
401	442
EDITED_BUSINESS_MESSAGE	emoji (aiogram.methods.send_sticker.SendSticker
(aiogram.enums.update_type.UpdateType	attribute), 368
attribute), 552	emoji (aiogram.types.dice.Dice attribute), 130
edited_business_message	emoji (aiogram.types.reaction_type_emoji.ReactionTypeEmoji
(aiogram.types.update.Update attribute),	attribute), 265
354	emoji (aiogram.types.sticker.Sticker attribute), 342
EDITED_CHANNEL_POST	<pre>emoji_list(aiogram.methods.set_sticker_emoji_list.SetStickerEmojiList</pre>
(aiogram.enums.update_type.UpdateType	emoji_list (aiogram.types.input_sticker.InputSticker
attribute), 552	attribute), 341
<pre>edited_channel_post (aiogram.types.update.Update</pre>	emoji_status_custom_emoji_id
auribuie), 333 EDITED_MESSAGE (aiogram.enums.update_type.UpdateTyp	
attribute), 552	attribute), 488
edited_message (aiogram.types.update.Update at-	emoji_status_custom_emoji_id
tribute), 353	(aiogram.types.chat.Chat attribute), 35
EditForumTopic (class in	emoji_status_custom_emoji_id
aiogram.methods.edit_forum_topic), 402	(aiogram.types.chat_full_info.ChatFullInfo
EditGeneralForumTopic (class in	attribute), 55
$aiogram.methods.edit_general_forum_topic),$	emoji_status_expiration_date
404	(aiogram.methods.set_user_emoji_status.SetUserEmojiStatus attribute), 488
EditMessageCaption (class in	emoji_status_expiration_date
aiogram.methods.edit_message_caption), 499	(aiogram.types.chat.Chat attribute), 35
EditMessageLiveLocation (class in	emoji_status_expiration_date
aiogram.methods.edit_message_live_location),	(aiogram.types.chat_full_info.ChatFullInfo
500	attribute), 55
EditMessageMedia (class in	EncryptedCredentials (class in
aiogram.methods.edit_message_media), 503	aiogram.types.encrypted_credentials), 344
EditMessageReplyMarkup (class in	EncryptedPassportElement (class in
aiogram.methods.edit_message_reply_markup),	aiogram.enums.encrypted_passport_element), 545

EncryptedPassportElement	(class	in	at	tribute), 548		
aiogram.types.encrypted_p 344	passport_elemen	t),	_	eBlockQuote ogram.utils.forma	(class tting), 658	in
enter() (aiogram.fsm.scene.Scene 610	esManager meth	nod),		n_date (<i>aiogram.:</i> tribute), 50	types.chat_boost.Ch	atBoost
enter() (aiogram.fsm.scene.SceneV				_	types.prepared_inlir	ne_message.PreparedInli
entities (aiogram.methods.edit_m	essage_text.Editi	Messa	-	tribute), 325		
attribute), 506 entities (aiogram.methods.send_n	nessage.SendMes	ssage	at	tribute), 391		te_link.CreateChatInvite
attribute), 452					ods.edit_chat_invite_	_link.EditChatInviteLink
entities (aiogram.types.input_text	_message_conte	nt.Inpi	-		aland invite limb Ch	
attribute), 324 entities (aiogram.types.message.	Massaga attrib	uta)		te (aiogram.types. tribute), 58	chat_invite_link.Ch	atinviteLink
202	-		explanati	on (aiogram.meth	ods.send_poll.Sendi	Poll at-
entities (aiogram.types.text_qu	uote.TextQuote	at-		ibute), 461	11 D 11 11	262
tribute), 270	anguar nya aha	akaut			.poll.Poll attribute),	263
error_message (aiogram.methods.attribute), 520	answer_pre_cnec	скоит_		iogram.methods.se	•	at
error_message(aiogram.methods.o	answer shinning	auer			гна_рон.ѕенағон	at-
attribute), 521	unswer_snipping	_quer		on_entities	(aiogram.types.p	oll Poll
ErrorEvent (class in aiogram.types	s.error_event). 6	17	_	tribute), 263	(atogram.types.p	ou.i ou
errors (aiogram.methods.set_passp						
attribute), 536			_	iogram.methods.se	end poll.SendPoll	at-
ETB (aiogram.enums.currency.Curre	ency attribute), 54	42		ibute), 461	- 4	
EUR (aiogram.enums.currency.Curre			export()(aiogram.utils.keyb	oard.InlineKeyboar	dBuilder
event (aiogram.types.update.Updat	e property), 355		m	ethod), 637		
event_business_connection_id			export()(aiogram.utils.keyb	oard.ReplyKeyboard	dBuilder
(aiogram.dispatcher.middl	ewares.data.Mid	ldlewa		ethod), 638		
attribute), 570			-	vite_link()	(aiogram.types.ch	at.Chat
event_chat (aiogram.dispatcher.mi	iddlewares.data.l	Middle				
attribute), 570			_	tInviteLink	(class	in
event_context (aiogram.dispatche attribute), 570			40)5	•	
event_from_user(aiogram.dispate	cher.middlewares	s.data.			types.message.Mess	age at-
attribute), 570	. 1 11	. 14.		ibute), 201	(1	
event_router (aiogram.dispatcher attribute), 570	.miaaiewares.aai	ta.wna			(class nal_reply_info), 131	in
event_thread_id(aiogram.dispate	char middlawara	s data				
attribute), 570	mer.maaiewares	s.aaia.	1 re micu enew Leu 62		е июдтит.иізриісне	n.jiugs),
event_type (aiogram.types.updat	e.Update prope	erty),	extract_f		pes.message_entity.	MessageEntity
event_update(aiogram.dispatcher.	middlewares.da	ta Mid		, · ·	osition_point.Mask)	PositionPoint
attribute), 570			_	tribute), 547		
exception (aiogram.types.error_e tribute), 617	event.ErrorEvent	at-	F	,,		
ExceptionMessageFilter	(class	in	FATI ED (aic	aram anume ravar	nue withdrawal star	te_type.RevenueWithdrav
aiogram.filters.exception),	581		at	tribute), 550		
exceptions (aiogram.filters.exceptions), 581	ion.ExceptionTyp	eFilte		update() (aiogra ethod), 565	m.dispatcher.dispate	cher.Dispatcher
ExceptionTypeFilter	(class	in			patcher.dispatcher.I	Dispatcher
aiogram.filters.exception),	*			ethod), 566	раненен амриненен.1	- ispaicine i
exit() (aiogram.fsm.scene.SceneW		12			assport element er	ror_data_field.Passport
EXPANDABLE_BLOCKQUOTE				tribute), 347	1	~
(aiogram.enums.message	entity type.Mess	ageEn		**		

file (aiogram.client.telegram.TelegramAPIServer	tribute), 262
attribute), 14 FILE(aiogram.enums.passport_element_error_type.Passport_element_element_error_type.Passport_element_ele	file_size (aiogram.types.sticker.Sticker attribute), 342 off Ele n shtær(aiðypa m.types.video.Video attribute), 274
attribute), 549	file_size (aiogram.types.video_note.VideoNote at-
File (class in aiogram.types.file), 133 file_date (aiogram.types.passport_file.PassportFile at-	tribute), 276 file_size (aiogram.types.voice.Voice attribute), 276
tribute), 353	file_unique_id (aiogram.types.animation.Animation
file_hash(aiogram.types.passport_element_error_file.Pa	assportElen ustnilbrno) File
attribute), 347 file_hash(aiogram.types.passport_element_error_front_	file_unique_id (aiogram.types.audio.Audio attribute),
attribute), 348	file_unique_id (aiogram.types.document.Document
${\tt file_hash} ({\it aiogram.types.passport_element_error_revers})$	
attribute), 349	file_unique_id (aiogram.types.file.File attribute), 133
attribute), 350	PfrislporufilapaentEdraniSglfaen.types.passport_file.PassportFile attribute), 352
file_hash(aiogram.types.passport_element_error_transl	
attribute), 350	attribute), 262
file_hashes (aiogram.types.passport_element_error_file attribute), 348	s .HasepuntEipenentErrojulitigs am.types.sticker.Sticker at- tribute), 341
file_hashes (aiogram.types.passport_element_error_transport_element_element_error_transport_element_el	· · · · · · · · · · · · · · · · · · ·
attribute), 351	274
file_id (aiogram.methods.get_file.GetFile attribute), 413	file_unique_id (aiogram.types.video_note.VideoNote attribute), 276
<pre>file_id (aiogram.types.animation.Animation attribute),</pre>	file_unique_id (aiogram.types.voice.Voice attribute), 276
file_id (aiogram.types.audio.Audio attribute), 20	file_url() (aiogram.client.telegram.TelegramAPIServer
file_id (aiogram.types.document.Document attribute), 130	method), 14 FILES (aiogram.enums.passport_element_error_type.PassportElementErro
file_id (aiogram.types.file.File attribute), 133	attribute), 549
<pre>file_id (aiogram.types.passport_file.PassportFile at- tribute), 352</pre>	files (aiogram.types.encrypted_passport_element.EncryptedPassportElement.incryptedPassportElemen
<pre>file_id (aiogram.types.photo_size.PhotoSize attribute),</pre>	fill (aiogram.types.background_type_fill.BackgroundTypeFill attribute), 23
file_id (aiogram.types.sticker.Sticker attribute), 341	fill (aiogram.types.background_type_pattern.BackgroundTypePattern
file_id (aiogram.types.video.Video attribute), 274 file_id (aiogram.types.video_note.VideoNote at-	attribute), 23 Filter (class in aiogram.filters.base), 582
tribute), 276	filter() (aiogram.filters.callback_data.CallbackData
file_id (aiogram.types.voice.Voice attribute), 276	class method), 579
file_name (aiogram.types.animation.Animation at- tribute), 20	FIND_LOCATION (aiogram.enums.chat_action.ChatAction attribute), 538
file_name (aiogram.types.audio.Audio attribute), 20	find_location() (aiogram.utils.chat_action.ChatActionSender
file_name (aiogram.types.document.Document at- tribute), 131	class method), 644 first_name(aiogram.methods.send_contact.SendContact
file_name (aiogram.types.video.Video attribute), 274	attribute), 439
file_path (aiogram.types.file.File attribute), 133	first_name (aiogram.types.chat.Chat attribute), 33
file_size (aiogram.types.animation.Animation attribute), 20	first_name (aiogram.types.chat_full_info.ChatFullInfo attribute), 55
file_size (aiogram.types.audio.Audio attribute), 20 file_size (aiogram.types.document.Document at-	first_name (aiogram.types.contact.Contact attribute), 129
tribute), 131	$\verb first_name (a iogram. types. in line_query_result_contact. In line Query Result_contact. I$
file_size (aiogram.types.file.File attribute), 133	attribute), 301
file_size (aiogram.types.passport_file.PassportFile at- tribute), 352	first_name(aiogram.types.input_contact_message_content.InputContaction attribute), 318
file size (aiogram types photo size PhotoSize at-	

lineQueryRe.
t.InputVenue
* 11. 0
.InlineQuery.
4 4 7 4 7 7
tent.InputVer
ionPartnerTy
ioni anineni
llowed
ne
ges
ges
ges Message
ges
ges Message
ı

```
from_markup() (aiogram.utils.keyboard.InlineKeyboardBuilatee_short_name (aiogram.methods.send_game.SendGame
              class method), 637
                                                                                                   attribute), 517
from_markup() (aiogram.utils.keyboard.ReplyKeyboardBuiglalne_short_name (aiogram.types.callback query.CallbackQuery
                                                                                                   attribute), 32
              class method), 638
from_request (aiogram.types.write_access_allowed.WriteAgemes_sAllowed_name (aiogram.types.inline_query_result_game.InlineQuery}
              attribute), 277
                                                                                                   attribute), 303
from_url() (aiogram.fsm.storage.mongo.MongoStorage
                                                                                     GameHighScore
                                                                                                                                    (class
                                                                                                                                                                    in
              class method), 593
                                                                                                   aiogram.types.game_high_score), 357
from_url()
                        (aiogram.fsm.storage.redis.RedisStorage
                                                                                     GBP (aiogram.enums.currency.Currency attribute), 542
              class method), 592
                                                                                     GEL (aiogram.enums.currency.Currency attribute), 542
from_user(aiogram.handlers.callback_query.CallbackQueGENERMLerFORUM_TOPIC_HIDDEN
                                                                                                   (aiogram.enums.content_type.ContentType
             property), 631
                                                                                                   attribute), 541
from_user(aiogram.types.callback_query.CallbackQuery
             attribute), 31
                                                                                     general_forum_topic_hidden
from\_user(aiogram.types.chat\_join\_request.ChatJoinRequest
                                                                                                   (aiogram.types.message.Message
                                                                                                                                                        attribute),
              attribute), 58
                                                                                                   205
from_user(aiogram.types.chat_member_updated.ChatMenGENHRMatFORUM_TOPIC_UNHIDDEN
             attribute), 106
                                                                                                   (aiogram.enums.content_type.ContentType
from_user(aiogram.types.chosen inline result.ChosenInlineResult attribute), 541
              attribute), 278
                                                                                     general_forum_topic_unhidden
from_user (aiogram.types.inline_query.InlineQuery at-
                                                                                                   (aiogram.types.message.Message
                                                                                                                                                        attribute),
             tribute), 278
from_user (aiogram.types.message.Message attribute), GeneralForumTopicHidden
                                                                                                                                                                   in
                                                                                                                                             (class
                                                                                                   aiogram.types.general forum topic hidden),
from_user(aiogram.types.paid media purchased.PaidMediaPurchased
             attribute), 327
                                                                                     GeneralForumTopicUnhidden
from_user(aiogram.types.pre_checkout_query.PreCheckoutQuery aiogram.types.general_forum_topic_unhidden),
              attribute), 327
from_user (aiogram.types.shipping_query.ShippingQuery get() (aiogram.fsm.scene.SceneRegistry method), 610
                                                                                     get_administrators()
              attribute), 332
                                                                                                                                   (aiogram.types.chat.Chat
FRONT_SIDE (aiogram.enums.passport_element_error_type.PassportEthenthentErrorType
              attribute), 549
                                                                                     get_data() (aiogram.fsm.scene.SceneWizard method),
front_side(aiogram.types.encrypted_passport_element.EncryptedPassportElement
              attribute), 345
                                                                                     get_data()
                                                                                                              (aiogram.fsm.storage.base.BaseStorage
FSInputFile (class in aiogram.types.input file), 182,
                                                                                                   method), 594
                                                                                     get_flag() (in module aiogram.dispatcher.flags), 620
fsm_storage (aiogram.dispatcher.middlewares.data.Middleware118nMiddleware.II8nMiddleware.II8nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddleware118nMiddlew
              attribute), 570
                                                                                                   method), 642
FSMI18nMiddleware
                                                   (class
                                                                                    get_member() (aiogram.types.chat.Chat method), 41
              aiogram.utils.i18n.middleware), 641
                                                                                     get_member_count()
                                                                                                                                   (aiogram.types.chat.Chat
FSMStrategy (class in aiogram.fsm.strategy), 595
                                                                                                   method), 42
full_name (aiogram.types.chat.Chat property), 37
                                                                                     get_profile_photos()
                                                                                                                                   (aiogram.types.user.User
full_name (aiogram.types.user.User property), 271
                                                                                                   method), 271
                                                                                                               (aiogram.fsm.storage.base.BaseStorage
                                                                                     get_state()
G
                                                                                                   method), 594
                                                                                     get_url() (aiogram.types.message.Message method),
GAME
            (aiogram.enums.content type.ContentType
                                                                                                   252
              tribute), 540
GAME (aiogram.enums.inline_query_result_type.InlineQueryRetANA;) ebleGifts
                                                                                                                                        (class
                                                                                                                                                                   in
                                                                                                   aiogram.methods.get_available_gifts), 361
              attribute), 546
{\tt game}\,(aiogram.types.external\_reply\_info.ExternalReplyInfo}\,{\tt GetBusinessConnection}
                                                                                                                                            (class
                                                                                                                                                                   in
                                                                                                   aiogram.methods.get_business_connection),
              attribute), 132
game (aiogram.types.message.Message attribute), 203
                                                                                     GetChat (class in aiogram.methods.get chat), 409
Game (class in aiogram.types.game), 356
                                                                                     GetChatAdministrators
                                                                                                                                            (class
                                                                                                                                                                    in
```

aiogram.methods.get_chat_administrators), 410		<pre>gif_height (aiogram.types.inline_query_result_gif.InlineQueryResultGif</pre>
GetChatMember (class aiogram.methods.get_chat_member), 411	in	<pre>gif_url (aiogram.types.inline_query_result_gif.InlineQueryResultGif</pre>
GetChatMemberCount (class aiogram.methods.get_chat_member_count),	in	<pre>gif_width(aiogram.types.inline_query_result_gif.InlineQueryResultGif</pre>
412		$\verb"gift" (a iogram.types.transaction_partner_chat.TransactionPartnerChat") \\$
GetChatMenuButton (class	in	attribute), 336
aiogram.methods.get_chat_menu_button), 412		<pre>gift(aiogram.types.transaction_partner_user.TransactionPartnerUser</pre>
GetCustomEmojiStickers (class	in	Gift (class in aiogram.types.gift), 340
aiogram.methods.get_custom_emoji_stickers 362	;),	GIFT_CODE (aiogram.enums.chat_boost_source_type.ChatBoostSourceType attribute), 538
GetFile (class in aiogram.methods.get_file), 413		gift_id (aiogram.methods.send_gift.SendGift attribute),
GetForumTopicIconStickers (class	in	366
aiogram.methods.get_forum_topic_icon_stic	kers)	gifts (aiogram.types.gifts.Gifts attribute), 340
414		Gifts (class in aiogram.types.gifts), 340
GetGameHighScores (class aiogram.methods.get_game_high_scores),	in	${\tt GIVEAWAY}~(aiogram.enums.chat_boost_source_type.ChatBoostSourceType~attribute), 538$
516		GIVEAWAY (aiogram.enums.content_type.ContentType at-
GetMe (class in aiogram.methods.get_me), 415		tribute), 541
GetMyCommands (class aiogram.methods.get_my_commands), 415		giveaway (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 132
GetMyDefaultAdministratorRights (class		giveaway (aiogram.types.message.Message attribute),
aiogram.methods.get_my_default_administra	ıtor_	
416		Giveaway (class in aiogram.types.giveaway), 136
GetMyDescription (class aiogram.methods.get_my_description), 417	in	GIVEAWAY_COMPLETED (aiogram.enums.content_type.ContentType attribute), 541
GetMyName (class in aiogram.methods.get_my_nam418	ıe),	<pre>giveaway_completed (aiogram.types.message.Message</pre>
GetMyShortDescription (class	in	GIVEAWAY_CREATED (aiogram.enums.content_type.ContentType
<pre>aiogram.methods.get_my_short_description) 419</pre>		attribute), 541 giveaway_created (aiogram.types.message.Message
GetStarTransactions (class	in	attribute), 205
aiogram.methods.get_star_transactions), 526		<pre>giveaway_message (aiogram.types.giveaway_completed.GiveawayCompleted), 137</pre>
GetStickerSet (class	in	giveaway_message_id
aiogram.methods.get_sticker_set), 363 GetUpdates (class in aiogram.methods.get_update		(aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGi attribute), 52
532	,,	giveaway_message_id
GetUserChatBoosts (class	in	(aiogram.types.giveaway_winners.GiveawayWinners
aiogram.methods.get_user_chat_boosts), 419		attribute), 138 GIVEAWAY_WINNERS (aiogram.enums.content_type.ContentType
GetUserProfilePhotos (class	in	attribute), 541
aiogram.methods.get_user_profile_photos), 420		giveaway_winners (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 132
GetWebhookInfo (class	in	giveaway_winners (aiogram.types.message.Message
aiogram.methods.get_webhook_info), 533 GIF (aiogram.enums.inline_query_result_type.InlineQ	uerv	attribute), 205 ReinktTyggyCompleted (class in
attribute), 545		aiogram.types.giveaway_completed), 137
<pre>gif_duration(aiogram.types.inline_query_result_gi</pre>	f.Inli	nacyneaws aiogram.types.giveaway_created), 137
<pre>gif_file_id(aiogram.types.inline_query_result_cac</pre>	hed_	
attribute), 289		aiogram.types.giveaway_winners), 137

tribute), 35

GLOBAL_USER (aiogram.fsm.strategy.FSMStrategy attribute), 595	has_private_forwards (aiogram.types.chat_full_info.ChatFullInfo
google_place_id(aiogram.methods.send_venue.SendVen	
	has_protected_content (aiogram.types.chat.Chat at-
<pre>google_place_id (aiogram.types.inline_query_result_ven attribute), 314</pre>	nue.InlineQ ຫຍ່າງທີ່ຂອ ຸຣູທີ່ຕັ້ນVenue has_protected_content
<pre>google_place_id(aiogram.types.input_venue_message_c</pre>	content.Inp(a VegnusiMisgsergeKatnfedt_ info.ChatFullInfo
attribute), 324	attribute), 56
	has_protected_content
tribute), 273 google_place_type (aiogram.methods.send_venue.SendV	(aiogram.types.message.Message attribute), Venue 202
attribute), 464	has_public_winners (aiogram.types.giveaway.Giveaway
${\tt google_place_type} \ (a iogram.types.in line_query_result_types.in lin$	venue.Inlin æQnibuyRy eshd6Venue
	has_restricted_voice_and_video_messages
$\verb google_place_type (a iogram. types. input_venue_message) $	
	has_restricted_voice_and_video_messages
google_place_type (aiogram.types.venue.Venue attribute), 273	(aiogram.types.chat_full_info.ChatFullInfo attribute), 56
	has_spoiler (aiogram.methods.send_animation.SendAnimation
GREEN (aiogram.enums.topic_icon_color.TopicIconColor	attribute), 433
	has_spoiler (aiogram.methods.send_photo.SendPhoto
GROUP (aiogram.enums.chat_type.ChatType attribute),	attribute), 458
539	has_spoiler (aiogram.methods.send_video.SendVideo
GROUP_CHAT_CREATED (aiogram.enums.content_type.Conte	
	$\verb has_spoiler (a iogram. types. input_media_animation. Input Media Animation and An$
<pre>group_chat_created (aiogram.types.message.Message</pre>	attribute), 184
attribute), 204	$\verb has_spoiler (a iogram.types.input_media_photo.InputMediaPhoto $
GTQ (aiogram.enums.currency.Currency attribute), 542	attribute), 187
Н	has_spoiler(aiogram.types.input_media_video.InputMediaVideo attribute), 189
$\verb+handler+ (a iogram. d is patcher. middle wares. data. Middle wares and a constant of the c$	
attribute), 570	attribute), 35
	has_visible_history
611	(aiogram.types.chat_full_info.ChatFullInfo
has_aggressive_anti_spam_enabled	attribute), 57
(aiogram.types.chat.Chat attribute), 35	hash(aiogram.types.encrypted_credentials.EncryptedCredentials
has_aggressive_anti_spam_enabled	attribute), 344
(aiogram.types.chat_full_info.ChatFullInfo attribute), 56	hash (aiogram.types.encrypted_passport_element.EncryptedPassportElem attribute), 344
	hash (aiogram.utils.web_app.WebAppInitData attribute),
(aiogram.types.webhook_info.WebhookInfo	648
	${\tt HASHTAG} \ (a iogram.enums.message_entity_type.MessageEntityType$
has_hidden_members (aiogram.types.chat.Chat at-	attribute), 548
tribute), 35	HashTag (class in aiogram.utils.formatting), 656
has_hidden_members(aiogram.types.chat_full_info.Chatiatribute), 56	heading (aiogram.methods.edit_message_live_location.EditMessageLiveL attribute), 502
has_main_web_app (aiogram.types.user.User attribute),	heading (aiogram.methods.send_location.SendLocation
271	attribute), 447
has_media_spoiler(aiogram.tvpes.external reply info.F	headine (qiqqqm.types.inline_query_result_location.InlineQueryResultL
attribute), 132	attribute), 307
	${\tt heading} \ (aiogram.types.input_location_message_content.InputLocationM$
attribute), 203	attribute), 323
has_private_forwards (aiogram.types.chat.Chat at- tribute) 35	heading (aiogram.types.location.Location attribute), 197

```
height (aiogram.methods.send_animation.SendAnimation I18nData (class in aiogram.dispatcher.middlewares.data),
              attribute), 433
                                                                                                    570
              (aiogram.methods.send video.SendVideo at- I18nMiddleware
                                                                                                                                       (class
                                                                                                    aiogram.utils.i18n.middleware), 642
              tribute), 467
height (aiogram.types.animation.Animation attribute), icon_color(aiogram.methods.create_forum_topic.CreateForumTopic
                                                                                                    attribute), 394
height (aiogram.types.input media animation.InputMedia Acomatiolior (aiogram.types.forum topic.ForumTopic at-
                                                                                                    tribute), 134
              attribute), 184
\label{lem:height} height (a iogram. types. input\_media\_video. InputMediaVide \verb"ocon_color" (a iogram. types. forum\_topic\_created. ForumTopicCreated) and the properties of 
              attribute), 189
                                                                                                    attribute), 135
height (aiogram.types.input_paid_media_video.InputPaidMedim_Videctom_emoji_id
              attribute), 190
                                                                                                    (aiogram.methods.create_forum_topic.CreateForumTopic
height (aiogram.types.paid_media_preview.PaidMediaPreview
                                                                                                    attribute), 394
                                                                                      icon_custom_emoji_id
              attribute), 261
height (aiogram.types.photo_size.PhotoSize attribute),
                                                                                                    (aiogram.methods.edit_forum_topic.EditForumTopic
                                                                                                    attribute), 403
height (aiogram.types.sticker.Sticker attribute), 342
                                                                                      icon_custom_emoji_id
height (aiogram.types.video.Video attribute), 274
                                                                                                    (aiogram.types.forum_topic.ForumTopic
HIDDEN_USER (aiogram.enums.message_origin_type.MessageOriginTyperibute), 134
                                                                                      icon_custom_emoji_id
              attribute), 548
hide_url (aiogram.types.inline_query_result_article.InlineQueryResultAgtionetypes.forum_topic_created.ForumTopicCreated
              attribute), 282
                                                                                                    attribute), 135
HideGeneralForumTopic
                                                                                      icon_custom_emoji_id
                                                       (class
              aiogram.methods.hide general forum topic),
                                                                                                    (aiogram.types.forum topic edited.ForumTopicEdited
              421
                                                                                                    attribute), 135
HKD (aiogram.enums.currency.Currency attribute), 542
                                                                                      id(aiogram.types.business\_connection.BusinessConnection
HNL (aiogram.enums.currency.Currency attribute), 542
                                                                                                    attribute), 29
horizontal_accuracy
                                                                                      id (aiogram.types.callback_query.CallbackQuery at-
              (aiogram.methods.edit_message_live_location.EditMessage_tiibalto);abibn
                                                                                      id (aiogram.types.chat.Chat attribute), 33
              attribute), 501
horizontal_accuracy
                                                                                      id (aiogram.types.chat_full_info.ChatFullInfo attribute),
              (aiogram.methods.send\_location.SendLocation
                                                                                      id (aiogram.types.gift.Gift attribute), 340
              attribute), 447
                                                                                      id (aiogram.types.inline_query.InlineQuery attribute),
horizontal_accuracy
              (aiogram.types.inline_query_result_location.InlineQueryRe3dBLocation
              attribute), 307
                                                                                      id(aiogram.types.inline_query_result_article.InlineQueryResultArticle
horizontal_accuracy
                                                                                                    attribute), 281
              (aiogram.types.input_location_message_content.lind(tliogationnt)\( \text{Messaighi@nqear} y_result_audio.InlineQueryResultAudio
              attribute), 322
                                                                                                    attribute), 282
horizontal_accuracy
                                                                                      id(aiogram.types.inline_query_result_cached_audio.InlineQueryResultCa
              (aiogram.types.location.Location
                                                                    attribute),
                                                                                                    attribute), 284
                                                                                      id(aiogram.types.inline_query_result_cached_document.InlineQueryResult_
HRK (aiogram.enums.currency.Currency attribute), 542
                                                                                                    attribute), 287
HTML (aiogram.enums.parse_mode.ParseMode attribute),
                                                                                      id(aiogram.types.inline_query_result_cached_gif.InlineQueryResultCache
                                                                                                    attribute), 288
html_text (aiogram.types.message.Message property),
                                                                                      id(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResu
                                                                                                    attribute), 291
                                                                                      id(aiogram.types.inline_query_result_cached_photo.InlineQueryResultCa
HUF (aiogram.enums.currency.Currency attribute), 542
                                                                                                    attribute), 294
                                                                                      id(aiogram.types.inline_query_result_cached_sticker.InlineQueryResultCa
                                                                                                    attribute), 296
i18n
            (aiogram.dispatcher.middlewares.data.I18nData
                                                                                      id(aiogram.types.inline_query_result_cached_video.InlineQueryResultCac
              attribute), 571
i18n_middleware(aiogram.dispatcher.middlewares.data.I18nData attribute), 298
                                                                                      id(aiogram.types.inline query result cached voice.InlineQueryResultCac
```

attribute), 571

```
attribute), 299
                                                       inline_message_id(aiogram.methods.edit_message_reply_markup.Editl
id (aiogram.types.inline_query_result_contact.InlineQueryResultContatttibute), 505
         attribute), 301
                                                       inline_message_id(aiogram.methods.edit message text.EditMessageTe.
id (aiogram.types.inline_query_result_document.InlineQueryResultDatwibente), 506
         attribute), 302
                                                       inline_message_id(aiogram.methods.get_game_high_scores.GetGameF
id (aiogram.types.inline query result game.InlineQueryResultGame attribute), 516
         attribute), 303
                                                       inline_message_id(aiogram.methods.set game score.SetGameScore
id(aiogram.types.inline_query_result_gif.InlineQueryResultGif
                                                                attribute), 519
         attribute), 304
                                                       inline_message_id(aiogram.methods.stop_message_live_location.Stop)
id (aiogram.types.inline_query_result_location.InlineQueryResultLocattioihute), 508
         attribute), 306
                                                       inline\_message\_id(aiogram.types.callback\_query.CallbackQuery)
id (aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMapacithGei), 31
                                                       inline_message_id(aiogram.types.chosen_inline_result.ChosenInlineRe
         attribute), 310
id (aiogram.types.inline_query_result_photo.InlineQueryResultPhotoattribute), 278
         attribute), 311
                                                       inline_message_id(aiogram.types.sent_web_app_message.SentWebApp
id(aiogram.types.inline_query_result_venue.InlineQueryResultVenueattribute), 325
         attribute), 313
                                                       INLINE_QUERY (aiogram.enums.update_type.UpdateType
id(aiogram.types.inline_query_result_video.InlineQueryResultVideo attribute), 552
         attribute), 315
                                                       inline_query (aiogram.types.update.Update attribute),
\verb"id" (a iogram.types.in line\_query\_result\_voice. In line QueryResult Voice~354
         attribute), 317
                                                       inline_query_id(aiogram.methods.answer_inline_query.AnswerInlineQ
id (aiogram.types.poll.Poll attribute), 262
                                                                attribute), 510
(class
                                                                                                          in
         attribute), 327
                                                                 aiogram.utils.keyboard), 636
\verb"id" (a iogram.types.prepared_in line\_message.Prepared In line\_\texttt{Messine} \texttt{K} eyboard \texttt{Button}
                                                                                          (class
                                                                                                          in
         attribute), 325
                                                                aiogram.types.inline_keyboard_button), 181
id (aiogram.types.shipping_option.ShippingOption at-
                                                       InlineKeyboardMarkup
                                                                                          (class
                                                                                                          in
         tribute), 331
                                                                aiogram.types.inline_keyboard_markup),
id
    (aiogram.types.shipping_query.ShippingQuery
         tribute), 332
                                                       InlineQuery (class in aiogram.types.inline_query), 278
id (aiogram.types.star_transaction.StarTransaction at-
                                                       InlineQueryResult
                                                                                        (class
         tribute), 333
                                                                 aiogram.types.inline_query_result), 279
id (aiogram.types.story.Story attribute), 269
                                                       InlineQueryResultArticle
                                                                                                          in
id (aiogram.types.user.User attribute), 270
                                                                aiogram.types.inline_query_result_article),
id (aiogram.utils.web app.WebAppChat attribute), 648
                                                                 280
id (aiogram.utils.web_app.WebAppUser attribute), 648
                                                       InlineQueryResultAudio
                                                                                            (class
                                                                                                          in
IDENTITY_CARD (aiogram.enums.encrypted_passport_element.EncrypticdPassprope&lintievet_query_result_audio),
         attribute), 545
IDR (aiogram.enums.currency.Currency attribute), 542
                                                       InlineQueryResultCachedAudio
                                                                                               (class
                                                                                                           in
ILS (aiogram.enums.currency.Currency attribute), 542
                                                                aiogram.types.inline_query_result_cached_audio),
InaccessibleMessage
         aiogram.types.inaccessible_message), 138
                                                       InlineQueryResultCachedDocument
include_router() (aiogram.dispatcher.router.Router
                                                                aiogram.types.inline_query_result_cached_document),
         method), 559
                                                                 285
include_routers() (aiogram.dispatcher.router.Router
                                                       InlineQueryResultCachedGif
                                                                                              (class
         method), 559
                                                                 aiogram.types.inline_query_result_cached_gif),
inline_keyboard(aiogram.types.inline_keyboard_markup.InlineKeyB&ardMarkup
         attribute), 182
                                                       InlineQueryResultCachedMpeg4Gif
                                                                                                 (class
                                                                                                           in
inline_message_id (aiogram.methods.edit_message_caption.EditMaissgram.appiasninline_query_result_cached_mpeg4_gif),
         attribute), 499
inline_message_id (aiogram.methods.edit_message_live_InhaineQuiatiyMessukgeCincHedPhiorbo
                                                                                               (class
                                                                                                          in
         attribute), 501
                                                                aiogram.types.inline_query_result_cached_photo),
\verb|inline_message_id| (a iogram.methods.edit\_message\_media.EditMessageMedia|) |
         attribute), 503
                                                       InlineQueryResultCachedSticker
                                                                                                (class
                                                                                                           in
```

aiogram.types.inline_query_result_cached_si		
294		input_message_content
InlineQueryResultCachedVideo (class aiogram.types.inline_query_result_cached_v.	in video`	(aiogram.types.inline_query_result_cached_audio.InlineQueryRe), attribute), 285
296		input_message_content
InlineQueryResultCachedVoice (class	in	(aiogram.types.inline_query_result_cached_document.InlineQuer
aiogram.types.inline_query_result_cached_v		
298		input_message_content
	in	(aiogram.types.inline_query_result_cached_gif.InlineQueryResult_
aiogram.types.inline_query_result_contact),		attribute), 289
300		input_message_content
	in	(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQue
aiogram.types.inline_query_result_document		attribute), 291
301		input_message_content
InlineQueryResultGame (class	in	(aiogram.types.inline_query_result_cached_photo.InlineQueryRe
aiogram.types.inline_query_result_game),		attribute), 294
303		input_message_content
InlineQueryResultGif (class	in	(aiogram.types.inline_query_result_cached_sticker.InlineQueryR
aiogram.types.inline_query_result_gif), 304		attribute), 296
		input_message_content
aiogram.types.inline_query_result_location), 305		(aiogram.types.inline_query_result_cached_video.InlineQueryReattribute), 298
InlineQueryResultMpeg4Gif (class	in	input_message_content
aiogram.types.inline_query_result_mpeg4_gi		(aiogram.types.inline_query_result_cached_voice.InlineQueryRe.
308	3 //	attribute), 300
InlineQueryResultPhoto (class	in	input_message_content
aiogram.types.inline_query_result_photo),		(aiogram.types.inline_query_result_contact.InlineQueryResultCo
311		attribute), 301
InlineQueryResultsButton (class	in	input_message_content
<pre>aiogram.types.inline_query_results_button), 318</pre>		(aiogram.types.inline_query_result_document.InlineQueryResulti attribute), 303
InlineQueryResultType (class	in	input_message_content
aiogram.enums.inline_query_result_type),		(aiogram.types.inline_query_result_gif.InlineQueryResultGif
545		attribute), 305
InlineQueryResultVenue (class	in	input_message_content
aiogram.types.inline_query_result_venue),		(aiogram.types.inline_query_result_location.InlineQueryResultLo
313		attribute), 307
InlineQueryResultVideo (class	in	input_message_content
aiogram.types.inline_query_result_video),		(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResult
314		attribute), 310
InlineQueryResultVoice (class	in	input_message_content
<pre>aiogram.types.inline_query_result_voice), 316</pre>		(aiogram.types.inline_query_result_photo.InlineQueryResultPhot attribute), 312
input_field_placeholder		input_message_content
	at-	(aiogram.types.inline_query_result_venue.InlineQueryResultVenu
tribute), 134		attribute), 314
input_field_placeholder		input_message_content
	lvKev	vboardMar \aip gram.types.inline_query_result_video.InlineQueryResultVideo
attribute), 266		attribute), 316
input_message_content		input_message_content
		QueryResu(latergiche .types.inline_query_result_voice.InlineQueryResultVoice attribute), 318
input_message_content		InputContactMessageContent (class in
		_ \

 $(a iogram. types. in line_query_result_audio. In lineQueryResult Augham. types. in put_contact_message_content),$

InputNoiceMessageContent (class in aiogram.types.input_invoice_message_content), 319 InputLocationMessageContent (class in aiogram.types.input_location_message_content), 322 InputMedia (class in aiogram.types.input_media_nimation), 183 InputMediaAndio (class in aiogram.types.input_media_aimination), 183 InputMediaAndio (class in aiogram.types.input_media_aimination), 183 InputMediaDocument (class in aiogram.types.input_media_adocument), 186 InputMediaPhoto (class in aiogram.types.input_media_photo), 187 InputMediaType (class in aiogram.types.input_media_video), 187 InputMediaType (class in aiogram.types.input_media_video), 187 InputMediaPhoto (class in aiogram.types.input_media_photo), 189 InputPaidMediaPhoto (class in aiogram.types.input_paid_media_type), 546 InputHedidPhoto (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaPhoto (class in aiogram.types.input_paid_	318	attribute), 430
aiogram.types.input_mocice_message_content) 319 InputCocationNessageContent (class in aiogram.types.input_media_lost in aiogram.types.input_pedia_media_lost in aiogram.types.input_lost in aiogram.types.input_pedia_media_lost in aiogram.types.input_lost in aiog	<pre>InputFile (class in aiogram.types.input_file), 182</pre>	<pre>invite_link (aiogram.types.chat.Chat attribute), 35</pre>
invite_link (alogram.types.char_invite_link ChatInviteLink attribute), 37 InputHedia (class in alogram.types.input_media), 183 InputHedia (class in alogram.types.input_media), 183 InputHedia (class in alogram.types.input_media animation), 183 InputHedia (class in alogram.types.input.types.tempthedia (class in alogram.types.input.types.tempthe	InputInvoiceMessageContent (class in	<pre>invite_link(aiogram.types.chat_full_info.ChatFullInfo</pre>
InputLocationMessageContent (class in airribute), 57 InputMedia Acidass in aiogram.types.input_media), 183 InputMedia Antimation (class in aiogram.types.input_media antimation), 184 InputMedia Photo (class in aiogram.types.input_media_photo), 187 InputMedia Photo (class in aiogram.types.input_media_photo), 187 InputMedia Photo (class in aiogram.types.input_media_video), 187 InputMedia Photo (class in aiogram.types.input_media_video), 187 InputMedia Video (class in aiogram.types.input_paid_media_yteo), 189 InputPaidMedia Control (class in aiogram.types.input_paid_media_yteo), 189 InputPaidMedia Video (class in aiogram.types.input_paid_media_yteo), 180 InputPaidMedia Video (class in aiogram.types.paid_media_yteo), 180 InputPaidMedia Video (class in aiogram.types.paid_media_yteo), 180 Inpu	aiogram.types.input_invoice_message_content),	attribute), 56
aiogram.types.input_peachton_message_content), invite_link(aiogram.types.that_join_request_ChatJoin_request_ChatJoin_request_ChatJoin_request_ChatJoin_request_ChatJoin_request_chat_join_request_ChatJoin_request_chat_join_request	319	<pre>invite_link(aiogram.types.chat_invite_link.ChatInviteLink</pre>
attribute), \$8 InputHedia (class in aiogram.types.input_media_(class in aiogram.types.input_media_(aution)), 183 InputHedia (class in aiogram.types.input_media_(audio)), 184 InputHedia (class in aiogram.types.input_media_(audio)), 184 InputHedia (class in aiogram.types.input_media_(audio)), 185 InputHedia (class in aiogram.types.input_media_(audio)), 187 InputHedia (class in aiogram.types.input_media_(aution)), 189 InputPaidHedia (class in aiogram.types.input_paid_media_(aution)), 189 InputPaidHedia (aution), 189 InputPaidHedia (class in aiogram.types.input_paid_media_(aution)), 189 InputPaidHedia (aution), 189 Inpu	InputLocationMessageContent (class in	attribute), 57
InputMediaAnimation (class in alogram.types.input_media_cnimation), 183 InputMediaAnimation (class in alogram.types.input_media_cnimation), 183 InputMediaDocument (class in alogram.types.input_media_coment), 186 InputMediaDocument (class in alogram.types.input_media_document), 186 InputMediaPhoto (class alogram.types.input_media_type), 546 InputMediaYideo (class in alogram.types.input_paid media_type), 546 InputPaidMedia (class in alogram.types.input_paid media_type), 546 InputPaidMediaYideo (class in alogram.types.input_paid_media_type), 546 InputPaidMediaYideo (class in alogram.types.input_paid_types.types		
InputHediaAnimation (class in aiogram.types.input_media_animation), 183 InputHediaAnudio (class in aiogram.types.input_media_animation), 184 InputHediaDocument (class in aiogram.types.input_media_aphoto), 187 InputHediaType (class in aiogram.types.input_media_photo), 187 InputHediaType input_media_photo), 187 InputHediaType input_media_photo), 187 InputHediaType india_media_vieo), 187 InputHediaType (class in aiogram.types.input_paid_media_photo), 189 InputFaidMediaType (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaType (class in	<pre>InputMedia (class in aiogram.types.input_media), 183</pre>	
InputRediaAudio (class in aiogram.types.input_media_audio), 184 InputRediaDocument (class in aiogram.types.input_media_document), 185 InputRediaPhoto (class in aiogram.types.input_media_photo), 187 InputRediaType (class in aiogram.types.mosage_media_photo), 187 InputRediaType (class in aiogram.types.input_media_photo), 187 InputRediaType (class in aiogram.types.input_media_photo), 187 InputRediaPhoto (class in aiogram.types.input_media_photo), 187 InputRediaPhoto (class in aiogram.types.input_media_photo), 187 InputRediaPhoto (class in aiogram.types.input_media_photo), 189 InputPaidMedia Photo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.types.input_paid_media_type), 189 InputPaidMediaType (class in aiogram.types.input_paid_media_type), 190 InputPaidMediaType (class in aiogram.types.		
ImputhediaAudio (class in aiogram.types.imput_media_audio), 184 InputhediaDocument (class in aiogram.types.imput_media_document), 186 InputhediaPhoto (class in aiogram.types.imput_media_photo), 187 InputhedialType (class in aiogram.types.imput_media_photo), 187 InputhedialType (class in aiogram.types.imput_media_photo), 187 InputhedialPhoto (class in aiogram.types.imput_paid_media_photo), 189 InputhedidialPhoto (class in aiogram.types.imput_paid_media_photo), 189 InputhedidialPhoto (class in aiogram.types.imput_paid_media_type), 189 InputhedidialPhoto (class in aiogram.types.imput_paid_media_type), 190 InputhedidialPhoto (class in aiogram.types.imput_paid_media_type), 190 InputhedialPhoto (class in aiogram.types.imput_paid_media_type), 1	aiogram.types.input_media_animation), 183	INVOICE (aiogram.enums.content_type.ContentType at-
InputMediaPhoto (class in aiogram.types.input_media_document), 186 invoice (aiogram.types.input_media_document), 186 invoice (aiogram.types.message_dessage_attribute), 204 invoice_payload (aiogram.types.mvoice), 326 invoice_payload (aiogram.types.mvoice), 326 invoice_payload (aiogram.types.mvoice), 326 invoice_payload (aiogram.types.mpes.mput_media_yideo), 187 invoice_payload (aiogram.types.mping_query.ShippingQuery_airribute), 328 invoice_payload (aiogram.types.mping_query.ShippingQuery_airribute), 332 invoice_payload (aiogram.types.mping_query.ShippingQuery_airribute), 333 invoice_payload (aiogram.types.mping_query.Shipping_query_shipp		tribute), 540
inputHediaPhoto (class aiogram.types.input_media_photo), 187 InputMediaType (class aiogram.types.input_media_photo), 187 InputMediaType (class aiogram.types.input_media_photo), 187 InputMediaType (class aiogram.types.input_media_video), 187 InputMediaType (class aiogram.types.input_media_video), 187 InputMediaType (class aiogram.types.input_paid_media), 189 InputPaidMedia Photo (class aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class aiogram.types.input_paid_media_type), 546 InputPaidMediaType (class aiogram.types.input_paid_media_tydeo), 190 InputPoino (class aiogram.types.input_paid_media_tydeo), 190 InputTextMessageContent (class aiogram.types.input_paid_media_tydeo)	aiogram.types.input_media_audio), 184	<pre>invoice(aiogram.types.external_reply_info.ExternalReplyInfo</pre>
InputMediaPhoto (class aiogram.types.input_media_photo), 187 InputMediaType (class aiogram.types.input_media_photo), 187 InputMediaType (class aiogram.types.input_media_video), 187 InputMediaVideo (class aiogram.types.input_media_video), 187 InputMessageContent (class aiogram.types.input_media_video), 187 InputPaidMedia Photo (class aiogram.types.input_paid_media_photo), 189 InputPaidMediaPhoto (class aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class aiogram.types.input_paid_media_video), 190 InputPollOption (class aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.transaction_partner_user.TransactionPartner_user.TransactionPartner_user.TransactionPartner_user.TransactionPartner_user.Trans	InputMediaDocument (class in	attribute), 133
InputMediaType (class in aiogram.types.input_media_type), 546 InputMediaVideo (class in aiogram.types.input_media_type), 546 InputMessageContent (class in aiogram.types.input_message_content), 323 InputPaidMedia (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaVideo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaVideo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 546 InputPatticker (class in aiogram.types.input_paid_media_type), 340 InputPattMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 INR (aiogram.enums.currency.Currency attribute), 542 Intensity (aiogram.num.enums.currency.Currency attribute), 542 Intensity (aiogram.num.enum.enums.encrypted_passport_idenum.enums.encrypted_passport_idenum.enums.encrypted_passport_idenum.enums.encrypted_passport_idenum.enum.encersge_matribute), 320 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_idenum.enum.encersge_matribute), 320 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_idenum.enum.encersge_matribute), 320 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_idenum.enum.encersge_matribute), 320 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_idenum.enum.encersge_matribute), 400 Invite_link (aiogram.methods.edit_chat_invite_link.EditChallnvitediinkynm.types.message_matribute), 402 Invite_link (aiogram.methods.edit_chat_invite_link.EditChallnvitediinkynm.types.message.encerton attribute), 482 Invite_invite_invite_invite_invite_link.EditChallnvitediinkynm.types.message.encerton attribute), 482 Invite_invite_invite_invite_invite_invite_invi	aiogram.types.input_media_document), 186	invoice (aiogram.types.message.Message attribute), 204
InputMediaType (class in aiogram.enums.input_media_video), 187 InputMessageContent (class in aiogram.types.input_media_video), 187 InputPaidMedia Photo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPollOption (class in aiogram.	InputMediaPhoto (class in	Invoice (class in aiogram.types.invoice), 326
InputMediaVideo (class in aiogram.types.input_paid_media_video), 187 InputPaidMediaVideo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaVideo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaVideo (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 190 InputPollOption (class in aiogram.types.input_paid_media_video), 190 InputPollOption (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_text_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.topat_text_message_content), 324 InputVenueMessageContent (class in aiogram.types.topat_text_message_content), 325 InputVenueMessageContent (class in aiogram.types.topat_text_message_content), 326 InputVenueMessageContent (class in aiogram.types.topat_text_message_content), 326	aiogram.types.input_media_photo), 187	<pre>invoice_payload(aiogram.types.pre_checkout_query.PreCheckoutQuery</pre>
InputMediaVideo (class in altribute), 329 invoice_payload (alogram.types.shipping_query.ShippingQuery attribute), 332 invoice_payload (alogram.types.shipping_query.ShippingQuery attribute), 332 invoice_payload (alogram.types.shipping_query.ShippingQuery attribute), 333 invoice_payload (alogram.types.successful_payment.SuccessfulPayment attribute), 334 invoice_payload (alogram.types.transaction_partner_user.Transaction_Pattribute), 339 invoice_payload (alogram.types.transaction_pattribute), 340 invoice_payload (alogram.types.sucker_bathcok.set/Webhook.set/Webhook.actribute), 339 invoice_payload (alogram.types.transaction_pattribute), 340 invoice_payload (alogram.types.transaction_pattribute), 340 invoice_payload (alogram.types.transaction_pattribute), 340 invoice_payload (alogram.types.sucker_bhook.actribute), 340 invoice_payload (alogram.types.	InputMediaType (class in	attribute), 328
InputNessageContent (class in aiogram.types.input_paid_media_tyleo), 189 InputPaidMedia (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.enums.input_paid_media_tyleo), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_tyleo), 190 InputPollOption (class in aiogram.types.input_paid_media_video), 190 InputPticker (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputPtitextMessageContent (class in aiogram.types.input_lext_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_lext_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 INR (aiogram.types.input_venue_message_content), 324 INR (aiogram.types.background_type_pattern.BackgissuaftSyperburstatiogram.types.chat_member_administrator.ChatMemberA attribute), 23 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_isteanoughedibassportpitespelitPoll attribute), 262 attribute), 400 invice_payload (aiogram.types.successful_payment.SuccessfulPayment attribute), 334 invoice_payload (aiogram.types.transaction_partner_user.TransactionPattribute), 339 invoice_payload (aiogram.types.transaction_partner_user.TransactionPattribute), 339 invoice_payload (aiogram.types.transaction_partner_user.TransactionPattribute), 339 invoice_payload (aiogram.types.transaction_partner_user.TransactionPattribute, 339 invoice_payload (aiogram.types.transaction_partner_user.Tr	aiogram.enums.input_media_type), 546	<pre>invoice_payload(aiogram.types.refunded_payment.RefundedPayment</pre>
InputNessageContent (class in alogram.types.input_message_content), 323 InputPaidMedia (class in alogram.types.input_paid_media, 189 InputPaidMediaPhoto (class in alogram.types.input_paid_media_photo), 189 InputPaidMediaPhoto (class in alogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in alogram.emums.imput_paid_media_type), 546 InputPaidMediaVideo (class in alogram.types.input_paid_media_video), 190 InputPaidMediaVideo (class in alogram.types.input_paid_media_video), 190 InputPollOption (class in alogram.types.input_poll_option), 191 InputSticker (class in alogram.types.input_poll_option), 191 InputSticker (class in alogram.types.input_poll_option), 191 InputPollOption (class in alogram.types.input_poll_option), 191 InputPollOption (class in alogram.types.input_poll_option), 191 InputSticker (class in alogram.types.input_poll_option), 193 InputPollOption (class in alogram.types.input_poll_option), 191 InputSticker	InputMediaVideo (class in	attribute), 329
invoice_payload (aiogram.types.successful_payment.SuccessfulPayment alignerm.types.input_paid_media, 189 InputPaidMediaPhoto (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaPhoto (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.enums.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.enums.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 545 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 324 INR (aiogram.types.input_venue_message_content), 324 INR (aiogram.enums.currency.Currency attribute), 542 Intensity (aiogram.types.background_type_pattern.BackgisuartipypetContent), 23 INTERNAL PASSPORT (aiogram.enums.encrypted_passport_ideamosytedbysaportyEtapoth Poll attribute), 262 Intribute), 340 Intribute), 345 Intribute), 340 Intribute), 345 Intribute),	aiogram.types.input_media_video), 187	<pre>invoice_payload(aiogram.types.shipping_query.ShippingQuery</pre>
InputPaidMedia (class in aiogram.types.input_paid_media), 189 InputPaidMediaPhoto (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPollOption (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputTextMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 325 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 326 InputVenueMessageContent (class in aiogram.types.chat_administrator_rights.ChatAdministrator_attribute), 48 Is_annonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_attribute), 48 Is_annonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_attribute), 49 Intensity (aiogram.types.background_type_pattern.BackgisupatiSpyribus (aiogram.types.chat_administrator_rights.ChatAdministrator_tribute), 49 Intensity (aiogram.types.background_type_pattern.BackgisupatiSpyribus (aiogram.types.chat_administrator_rights.ChatAdministrator_tribute), 49 Intensity (aiogram.types.background_type_pattern.BackgisupatiSpyribus (aiogram.types.chat_administrator_rights.ChatAdministra	InputMessageContent (class in	attribute), 332
InputPaidMedia (class in aiogram.types.input_paid_media), 189 InputPaidMediaPhoto (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.enums.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.enums.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_type), 190 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPollOption (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputTextMessageContent (class in aiogram.types.input_lext_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 325 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 326 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 327 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 329 InputVenueMessageContent (class in aiogram.types.toker_sticke	aiogram.types.input_message_content), 323	<pre>invoice_payload(aiogram.types.successful_payment.SuccessfulPayment</pre>
InputPaidMediaPhoto (class in aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.enums.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPollOption (class in aiogram.types.input_paid_option), 191 InputSticker (class in aiogram.types.input_paid_option), 191 InputSticker (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.chat_member_administrator_rights.ChatAdministrator_attribute), 23 InputVenueMessageContent (class in aiogram.types.background_type_pattern.BackgissuadiSpyritums(aiogram.types.chat_member_administrator.ChatMemberA_attribute), 23 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_isdeamonffonsys_teld@ssynryfisemethtPoll attribute), 262 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite_lainkgram.types.message.Message attribute), 201 invite_link (aiogram.methods.edit_chat_subscription_invite_link(link_initeration_tattribute), 402 attribute), 339 ip_address (aiogram.methods.set_webhook.set_webhoo		attribute), 334
aiogram.types.input_paid_media_photo), 189 InputPaidMediaType (class in aiogram.enums.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPl1Option (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_poll_option), 191 InputTextMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.totat_administrator_rights.ChatAdministrator_aiogram.types.input_venue_message_content), 324 InputVenueMessageContent (class in aiogram.types.chat_member_administrator_rights.ChatAdministrator_aiogram.types.chat_member_administrator.ChatMemberA attribute), 48 is_anonymous (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 23 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_idenanopmousp(eidigssaport)phenetic), 343 is_anonymous (aiogram.types.chat_member_administrator_rights.ChatAdministrator_attribute), 48 is_anonymous (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 542 is_anonymous (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 545 is_anonymous (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 545 is_anonymous (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 48 is_anonymous (aiogram.types.chat_m	aiogram.types.input_paid_media), 189	<pre>invoice_payload(aiogram.types.transaction_partner_user.TransactionPages.transaction_pages.transac</pre>
InputPaidMediaType (class in ip_address (aiogram.types.webhook_info.WebhookInfo aiogram.enums.input_paid_media_type), 546 ip_filter_middleware() (in module aiogram.types.input_paid_media_video), 190 ip_filter_middleware() (in module aiogram.webhook.aiohttp_server), 624 ip_filter_middleware() (in module aiogram.types.input_paid_media_video), 190 ip_filter_middleware() (in module aiogram.webhook.aiohttp_server), 624 ip_filter_middleware() (in module aiogram.types.tider_server), 624 ip_filter_middleware() (in module aiogram.types	InputPaidMediaPhoto (class in	attribute), 339
InputPaidMediaType (class in ip_address (aiogram.types.webhook_info.WebhookInfo		
aiogram.enums.input_paid_media_type), 546 InputPaidMediaVideo (class in aiogram.types.input_paid_media_video), 190 InputPollOption (class in aiogram.types.input_poll_option), 191 InputSticker (class in aiogram.types.input_sticker), 340 InputTextMessageContent (class in aiogram.types.input_text_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 323 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 INR (aiogram.enums.currency.Currency attribute), 542 Intensity (aiogram.types.background_type_pattern.Background_type_type_type_type_type_type_type_type	InputPaidMediaType (class in	
ip_filter_middleware() (in module InputPaidMediaVideo (class in aiogram.types.input_paid_media_video),	<pre>aiogram.enums.input_paid_media_type),</pre>	
aiogram.types.input_paid_media_video , 190 is_animated (aiogram.webhook.security), 624 is_animated (aiogram.types.sticker_sticker_attribute), 342 is_animated (aiogram.types.sticker_set.StickerSet attribute), 343 is_animated (aiogram.types.sticker_set.StickerSet attribute), 340 is_animated (aiogram.types.sticker_set.StickerSet attribute), 343 is_animated (aiogram.types.sticker_set.StickerSet attribute), 340 is_animated (aiogram.types.sticker_set.StickerSet attribute), 343 is_animated (aiogram.types.sticker_set.StickerSet attribute), 343 is_animated (aiogram.methods.promote_chat_member_PromoteChatMen attribute), 342 is_animated (aiogram.methods.promote_chat_member.PromoteChatMen attribute), 343 is_animated (aiogram.methods.promote_chat_member.PromoteChatMen attribute), 343 is_animated (aiogram.methods.promote_chat_member.PromoteChatMen attribute), 343 is_animated (aiogram.methods.promote_chat_member.PromoteChatMen attribute), 343 is_animated (aiogram.methods.promote_chat_member.PromoteChatMen attribute), 445 is_animated (aiogram.methods.promote_chat_member.PromoteChatMen attribute), 343 is_animated (aiogram.types.sticker_set.Sticker		<pre>ip_filter_middleware()</pre>
is_animated (aiogram.types.sticker.Sticker attribute), InputPollOption (class in aiogram.types.input_poll_option), 191 is_animated (aiogram.types.sticker_set.StickerSet at- InputSticker (class in aiogram.types.input_sticker), 340 is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMentitude), 425 InputVenueMessageContent (class in aiogram.types.input_text_message_content), 323 is_anonymous (aiogram.methods.send_poll.SendPoll attribute), 461 InputVenueMessageContent (class in aiogram.types.input_venue_message_content), 324 is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_aiogram.types.input_venue_message_content), 324 is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_aiogram.types.chat_member_administrator.ChatMemberA_attribute), 48 INR (aiogram.enums.currency.Currency attribute), 542 attribute), 101 intensity (aiogram.types.background_type_pattern.Backgiosunatiopynthous (aiogram.types.chat_member_administrator.ChatMemberOwner_attribute), 23 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_istermonfmonsycedforsproryfessporth.Poll attribute), 262 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite_linkgram.types.message.Message_attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_biatgChantSuthscriptsion_Imeistatigak_reaction.SetMessageReaction_attribute), 482	InputPaidMediaVideo (class in	aiogram.webhook.aiohttp_server), 624
InputPollOption (class in aiogram.types.input_poll_option), 191 is_animated (aiogram.types.sticker.Sticker attribute), 342 is_animated (aiogram.types.sticker_set.StickerSet at-InputSticker (class in aiogram.types.input_sticker), 343 is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMethods.agam.types.input_text_message_content), 323 is_anonymous (aiogram.methods.send_poll.SendPoll attribute), 461 is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_aiogram.types.input_venue_message_content), 324 is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_aiogram.types.background_type_pattern.Backgisusatiopypatous(aiogram.types.chat_member_administrator.ChatMemberA_attribute), 23 attribute), 101 intensity (aiogram.enums.encrypted_passport_isternonfmousp(edfbyssportyEtespotht.Poll attribute), 262 is_automatic_forward_invite_link(aiogram.methods.edit_chat_invite_link.EditChatInvite_linkgram.types.message.Message_attribute), 201 invite_link (aiogram.methods.edit_chat_subscription_invite_bing(EditigOuntSuthsuclpsion_Imvitesiafgak_reaction.SetMessageReaction_attribute), 402 attribute), 482	aiogram.types.input_paid_media_video),	IPFilter (class in aiogram.webhook.security), 624
aiogram.types.input_poll_option), 191 is_animated (aiogram.types.sticker_set.StickerSet at- InputSticker (class in aiogram.types.input_sticker),		<pre>is_animated (aiogram.types.sticker.Sticker attribute),</pre>
InputSticker (class in aiogram.types.input_sticker), 340 is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMen InputTextMessageContent (class in attribute), 425 aiogram.types.input_text_message_content), 323 is_anonymous (aiogram.methods.send_poll.SendPoll at- tribute), 461 InputVenueMessageContent (class in is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrat aiogram.types.input_venue_message_content), 324 is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrat aiogram.enums.currency.Currency attribute), 542 attribute), 101 intensity (aiogram.types.background_type_pattern.BackgissutariSpyrtflotts.taiogram.types.chat_member_owner.ChatMemberOwner attribute), 23 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_ieseranonflootsp(edibgssportyfilespoilt.Poll attribute), 262 attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInviteLaitkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_birg (EditgChoot.SuthIscodpsion_Invested.gak_reaction.SetMessageReaction attribute), 402	InputPollOption (class in	342
is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMethologymous) is_anonymous (aiogram.methods.promote_chat_member.PromoteChatMethologymous) aiogram.types.input_text_message_content), 323	aiogram.types.input_poll_option), 191	<pre>is_animated (aiogram.types.sticker_set.StickerSet at-</pre>
InputTextMessageContent (class in attribute), 425 aiogram.types.input_text_message_content), is_anonymous (aiogram.methods.send_poll.SendPoll at- 323	<pre>InputSticker (class in aiogram.types.input_sticker),</pre>	tribute), 343
aiogram.types.input_text_message_content), is_anonymous (aiogram.methods.send_poll.SendPoll at- 323	340	$\verb"is_anonymous" (a iogram.methods.promote_chat_member.PromoteChatMethods) and a support of the promote of the$
tribute), 461 InputVenueMessageContent (class in is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrator_aiogram.types.input_venue_message_content), attribute), 48 324 is_anonymous (aiogram.types.chat_member_administrator.ChatMemberA INR (aiogram.enums.currency.Currency attribute), 542 attribute), 101 intensity (aiogram.types.background_type_pattern.Backgissundfispyethatus (miogram.types.chat_member_owner.ChatMemberOwner attribute), 23 attribute), 104 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_isteranonfinosyp(edbyssportyfels)pohlt.Poll attribute), 262 attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite(ainkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_birty(EiitigGhantSudtsscription_Imvited_gink_reaction.SetMessageReaction attribute), 402 attribute), 482	InputTextMessageContent (class in	attribute), 425
aiogram.types.input_venue_message_content), attribute), 48 324 is_anonymous (aiogram.types.chat_member_administrator.ChatMemberA INR (aiogram.enums.currency.Currency attribute), 542 attribute), 101 intensity (aiogram.types.background_type_pattern.Backgicsuadibyperrate (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 23 attribute), 104 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_iesenanonspleadbgrssporyfelsmodd.Poll attribute), 262 attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvitelainkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_bird (EditgChantSudrkendption_Inveited.gink_reaction.SetMessageReaction attribute), 402 attribute), 482		
aiogram.types.input_venue_message_content), attribute), 48 324 is_anonymous (aiogram.types.chat_member_administrator.ChatMemberA INR (aiogram.enums.currency.Currency attribute), 542 attribute), 101 intensity (aiogram.types.background_type_pattern.BackgicsuadibyerRatts (aiogram.types.chat_member_owner.ChatMemberOwner	InputVenueMessageContent (class in	is_anonymous (aiogram.types.chat_administrator_rights.ChatAdministrat
is_anonymous (aiogram.types.chat_member_administrator.ChatMemberAtting (aiogram.enums.currency.Currency attribute), 542 intensity (aiogram.types.background_type_pattern.BackgicsuradicpyetRotts (niogram.types.chat_member_owner.ChatMemberOwner attribute), 23 attribute), 104 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_icseranonfmonsp(edfogssportyfilespenhl.Poll attribute), 262 attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite(ainkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_birty (EtitigChantSudtkodption_Inveited.gak_reaction.SetMessageReaction attribute), 402 attribute), 402 attribute), 482		•
intensity (aiogram.types.background_type_pattern.BackgicsupathSppethouts (niogram.types.chat_member_owner.ChatMemberOwner attribute), 23 attribute), 104 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_icse_nanon.Spnonsp(eathgrssport)Sespoils.Poll attribute), 262 attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite{ainkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_birds (EibigEdmont.Suchtkoodptsion_Inveited.gink_reaction.SetMessageReaction attribute), 402 attribute), 402		is_anonymous(aiogram.types.chat_member_administrator.ChatMemberA
attribute), 23 attribute), 104 INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_ieseranonEmonsplead@gssportElesnoollt.Poll attribute), 262 attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite(aidgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_binds (EibigEdmantSuch kood ptsion_Invested gnk_reaction.SetMessageReaction attribute), 402 attribute), 482	INR (aiogram.enums.currency.Currency attribute), 542	attribute), 101
INTERNAL_PASSPORT (aiogram.enums.encrypted_passport_ieseranonsfrmonsp(edbgrssporyfilespetht.Poll attribute), 262	$\verb intensity (a iogram. types. background_type_pattern. Background_ty$	
attribute), 545 is_automatic_forward invite_link (aiogram.methods.edit_chat_invite_link.EditChatInvite[ainkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_bink (EditgEdmantSudskoodpsien_Inveitedign_kreaction.SetMessageReaction attribute), 402 attribute), 482		
invite_link (aiogram.methods.edit_chat_invite_link.EditChatInviteLainkgram.types.message.Message attribute), attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_bird; (EditgEdmantSudskoodption_Invested gnkreaction.SetMessageReaction attribute), 402 attribute), 482		
attribute), 400 201 invite_link (aiogram.methods.edit_chat_subscription_invite_binds (EitigEdmant.Such kendptsion_linveitsed gink_reaction.SetMessageReaction attribute), 402 attribute), 482		
invite_link (aiogram.methods.edit_chat_subscription_invite_ birds (Eilit Cluont.Studikscrdptsion_Imveitsed gink reaction.SetMessageReaction attribute), 402 attribute), 482		
	$\verb"invite_link" (a iogram.methods.edit_chat_subscription_in") in the control of $	n vii se_ bird (Eili g&mont. Sudt kood pti on <u>I</u> mmeit val ga <u>k</u> reaction. Set Message Reaction

attribute), 24	attribute), 201
<pre>is_bot (aiogram.types.user.User attribute), 270 is_bot (aiogram.utils.web_app.WebAppUser attribute),</pre>	is_unclaimed(aiogram.types.chat_boost_source_giveaway.ChatBoostSouattribute), 52
648	is_video (aiogram.types.sticker.Sticker attribute), 342
is_canceled(aiogram.methods.edit_user_star_subscripting attribute), 525	
is_closed (aiogram.methods.send_poll.SendPoll	ISK (aiogram.enums.currency.Currency attribute), 543
attribute), 461	ITALIC (aiogram.enums.message_entity_type.MessageEntityType
is_closed (aiogram.types.poll.Poll attribute), 262	attribute), 548
is_disabled(aiogram.types.link_preview_options.LinkPrattribute), 196	
is_enabled(aiogram.types.business_connection.Business	
attribute), 29	JMD (aiogram.enums.currency.Currency attribute), 543
is_first_recurring(aiogram.types.successful_paymen attribute), 334	36
1s_flexible (aiogram.methods.create_invoice_link.Crea	te]BïqicBljinkequest (aiogram.types.chat_full_info.ChatFullInfo
attribute), 524	attribute), 56
is_flexible (aiogram.methods.send_invoice.SendInvoice attribute), 530	s join_to_send_messages (aiogram.types.chat.Chat at- tribute), 36
is_flexible (aiogram.types.input_invoice_message_cont	tribute), 50 tent InputInvoiceMessaceContent
attribute), 322	(aiogram.types.chat_full_info.ChatFullInfo
is_forum (aiogram.types.chat.Chat attribute), 33	attribute), 56
$\verb"is_forum" (a iogram. types. chat_full_info. ChatFullInfo~at-$	JPY (aiogram.enums.currency.Currency attribute), 543
tribute), 55	
is_from_offline (aiogram.types.message.Message at-	K
tribute), 202	KES (aiogram.enums.currency.Currency attribute), 543
1s_inverted (alogram.types.background_type_pattern.Ba	kes (alogram.enums.currency.Currency attribute), 545 akeroundTypePattern keyboard (alogram.types.reply_keyboard_markup.ReplyKeyboardMarkup
attribute), 23 is_local (aiogram.client.telegram.TelegramAPIServer	attribute), 266
attribute), 15	KeyboardButton (class in
is_manual (aiogram.types.text_quote.TextQuote at-	aiogram.types.keyboard_button), 191
<i>tribute</i>), 270	KeyboardButtonPollType (class in
is_member(aiogram.types.chat_member_restricted.ChatMattribute), 105	172
is_moving (aiogram.types.background_type_pattern.Back attribute), 23	KeyboardButtonPollTypeType (class in groundTypePattern.enums.keyboard_button_poll_type_type),
is_moving (aiogram.types.background_type_wallpaper.Ba	547 ackgroundTypeWallpaner
attribute), 24	"KeyboardButtonRequestChat (class in
attribute), 24 is_persistent (aiogram.types.reply_keyboard_markup.l	atogram.types.keyboara_button_request_cnat), ReplyKeyboardMarkup
attribute), 266	KeyboardButtonRequestUser (class in
is_personal (aiogram.methods.answer_inline_query.Ans	werInlineQuery merInlineQuery diogram.types.keyboard_button_request_user).
attribute), 510	194
is_premium (aiogram.types.user.User attribute), 271	KeyboardButtonRequestUsers (class in
is_premium (aiogram.utils.web_app.WebAppUser attribute), 648	aiogram.types.keyboard_button_request_users), 195
is_primary(aiogram.types.chat_invite_link.ChatInviteLin	nk KeyBuilder (class in aiogram.fsm.storage.base), 593
attribute), 57	keywords (aiogram.methods.set_sticker_keywords.SetStickerKeywords
1s_recurring (aiogram.types.successful_payment.Succes	keywords (aiogram.methods.set_sticker_keywords.SetStickerKeywords styllPayment attribute), 372
<pre>attribute), 334 is_revoked (aiogram.types.chat_invite_link.ChatInviteLin</pre>	keywords (aiogram.types.input_sticker.InputSticker at-
attribute), 57	1110the), 541
is_star_qiveaway(aiogram.tvpes.giveawav completed.	KGS (aiogram.enums.currency.Currency attribute), 543 Gireaway Completed KICKED (aiogram.enums.chat_member_status.ChatMemberStatus
attribute), 137	KICKEB (alogram.enums.cnat_member_status.ChatMemberStatus
<pre>is_topic_message (aiogram.types.message.Message</pre>	attribute), 539 KRW (ajogram enums currency Currency attribute), 543

KZT (aiogram.enums.currency.Currency attribute), 543	latitude (aiogram.methods.edit_message_live_location.EditMessageLiverattribute), 501
L	${\tt latitude} (a iogram. methods. send_location. Send Location$
label (aiogram.types.labeled_price.LabeledPrice attribute), 326	attribute), 447 latitude (aiogram.methods.send_venue.SendVenue at-
LabeledPrice (class in aiogram.types.labeled_price), 326	tribute), 463 latitude (aiogram.types.inline_query_result_location.InlineQueryResultL
language (aiogram.types.message_entity.MessageEntity attribute), 256	attribute), 306 latitude (aiogram.types.inline_query_result_venue.InlineQueryResultVenue.
language_code (aiogram.methods.delete_my_commands. attribute), 399	latitude(aiogram.types.input_location_message_content.InputLocationM
language_code (aiogram.methods.get_my_commands.Ge attribute), 415	tMyCommaft#sibute), 322 latitude(aiogram.types.input_venue_message_content.InputVenueMessa
language_code (aiogram.methods.get_my_description.Geattribute), 417	etMyDescrifttfibute), 324 latitude (aiogram.types.location.Location attribute),
language_code (aiogram.methods.get_my_name.GetMyN attribute), 418	Tame 196 LBP (aiogram.enums.currency.Currency attribute), 543
language_code (aiogram.methods.get_my_short_descript attribute), 419	
language_code (aiogram.methods.set_my_commands.Set. attribute), 483	MyecynChatusclass in aiogram.methods.leave_chat), 422 LEFT (aiogram.enums.chat_member_status.ChatMemberStatus
language_code (aiogram.methods.set_my_description.Set attribute), 485	
language_code (aiogram.methods.set_my_name.SetMyNo attribute), 486	7
language_code (aiogram.methods.set_my_short_descript attribute), 487	• • • • • • • • • • • • • • • • • • • •
language_code (aiogram.types.user.User attribute), 271	attribute), 469
language_code (aiogram.utils.web_app.WebAppUser attribute), 648	length (aiogram.types.message_entity.MessageEntity attribute), 256
last_error_date (aiogram.types.webhook_info.Webhook attribute), 355	k/hength (aiogram.types.video_note.VideoNote attribute), 276
last_error_message (aiogram.types.webhook_info.Webbattribute), 355	hdinnito(aiogram.methods.get_star_transactions.GetStarTransactions attribute), 526
last_name (aiogram.methods.send_contact.SendContact attribute), 439	limit (aiogram.methods.get_updates.GetUpdates at- tribute), 532
last_name (aiogram.types.chat.Chat attribute), 33	limit (aiogram.methods.get_user_profile_photos.GetUserProfilePhotos attribute), 420
last_name (aiogram.types.chat_full_info.ChatFullInfo attribute), 55	link_preview (aiogram.client.default.DefaultBotProperties
last_name (aiogram.types.contact.Contact attribute),	attribute), 557
129	link_preview_is_disabled
<pre>last_name (aiogram.types.inline_query_result_contact.In</pre>	attribute), 557
last_name (aiogram.types.input_contact_message_content attribute), 318	nt. <mark>lhpkrOGANieAMesNateleOSh</mark> tent (aiogram.methods.edit_message_text.EditMessageText
last_name (aiogram.types.shared_user.SharedUser at- tribute), 268	<pre>attribute), 506 link_preview_options</pre>
last_name (aiogram.types.user.User attribute), 270	(aiogram.methods.send_message.SendMessage
last_name (aiogram.utils.web_app.WebAppUser at-	attribute), 452
tribute), 648	link_preview_options
last_synchronization_error_date (aiogram.types.webhook_info.WebhookInfo	(aiogram.types.external_reply_info.ExternalReplyInfo attribute), 132
attribute), 355	link_preview_options (ajogram types input_text_message_content_InputTextMessageCo
	CHOSTON INDEX MONT JEXT MEXILIPE COMPUT MONTE EXIMPSSAOPLA

<pre>attribute), 324 link_preview_options</pre>	LogOut (class in aiogram.methods.log_out), 423 longitude (aiogram.methods.edit_message_live_location.EditMessageLive
(aiogram.types.message.Message attribute),	attribute), 501
202 link_preview_prefer_large_media	longitude (aiogram.methods.send_location.SendLocation attribute), 447
(aiogram.client.default.DefaultBotProperties attribute), 557	longitude (aiogram.methods.send_venue.SendVenue attribute), 463
link_preview_prefer_small_media	${\tt longitude} \ (a iogram. types. in line_query_result_location. In line Query Result_location. In line Query Result_locat$
(aiogram.client.default.DefaultBotProperties attribute), 557	attribute), 306 longitude (aiogram.types.inline_query_result_venue.InlineQueryResultVe
link_preview_show_above_text	attribute), 313
(aiogram.client.default.DefaultBotProperties attribute), 557	longitude (aiogram.types.input_location_message_content.InputLocation.attribute), 322
<pre>linked_chat_id (aiogram.types.chat.Chat attribute),</pre>	longitude (aiogram.types.input_venue_message_content.InputVenueMessage_ttribute), 324
<pre>linked_chat_id (aiogram.types.chat_full_info.ChatFullI</pre>	infloongitude (aiogram.types.location.Location attribute), 197
LinkPreviewOptions (class in	N.A.
aiogram.types.link_preview_options), 196	M
live_period(aiogram.methods.edit_message_live_locati	
attribute), 501	magic_data (aiogram.filters.magic_data.MagicData at-
live_period (aiogram.methods.send_location.SendLocat	
attribute), 447 live_period(aiogram.types.inline_query_result_location	magic_result(aiogram.filters.command.CommandObject
attribute), 307	MagicData (class in aiogram.filters.magic_data), 578
	าทุ ลเหมายหลองราง กา <mark>Meissoga Connem</mark> t session.base.BaseSession
	method), 15 MARKDOWN (aiogram.enums.parse_mode.ParseMode at-
tribute), 197	tribute), 549
LKR (aiogram.enums.currency.Currency attribute), 543 LOCATION (aiogram.enums.content_type.ContentType at-	MARKDOWN_V2 (aiogram.enums.parse_mode.ParseMode attribute), 549
tribute), 540	MASK (aiogram.enums.sticker_type.StickerType attribute),
LOCATION (aiogram.enums.inline_query_result_type.Inline	
attribute), 546	mask_position(aiogram.methods.set_sticker_mask_position.SetStickerMe
location (aiogram.types.business_location.BusinessLoca	
attribute), 30 location (aiogram.types.chat.Chat attribute), 36	mask_position(aiogram.types.input_sticker.InputSticker
location (aiogram.types.chat_full_info.ChatFullInfo at-	attribute), 341 mask_position (aiogram.types.sticker.Sticker attribute),
tribute), 57	342
	MaskPosition (class in aiogram.types.mask_position), 341
location (aiogram.types.chosen_inline_result.ChosenInliattribute), 278	
	olymania connections (aiogram.methods.set_webhook.SetWebhook attribute), 535
location (aiogram.types.inline_query.InlineQuery at- tribute), 278	max_connections (aiogram.types.webhook_info.WebhookInfo attribute), 356
location (aiogram.types.message.Message attribute), 203	max_quantity (aiogram.types.keyboard_button_request_users.KeyboardB attribute), 195
location (aiogram.types.venue.Venue attribute), 273	max_reaction_count(aiogram.types.chat_full_info.ChatFullInfo
Location (class in aiogram.types.location), 196	attribute), 55
	KandxoardButtaount (aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 524
LoginUrl (class in aiogram.types.login_url), 197	

```
max_tip_amount (aiogram.methods.send_invoice.SendInvoice
                                                                                                                                                                    method), 271
                       attribute), 529
                                                                                                                                            mentioned (aiogram.filters.command.CommandObject
max_tip_amount(aiogram.types.input_invoice_message_content.Inputbparity)MessageContent
                                                                                                                                            menu_button(aiogram.methods.set_chat_menu_button.SetChatMenuButto
                       attribute), 321
MaybeInaccessibleMessage
                                                                                              (class
                                                                                                                                                                    attribute), 476
                      aiogram.types.maybe inaccessible message),
                                                                                                                                            MenuButton (class in aiogram.types.menu button), 198
                                                                                                                                            MenuButtonCommands
                                                                                                                                                                                                                                  (class
                                                                                                                                                                                                                                                                              in
md_text (aiogram.types.message.Message property),
                                                                                                                                                                    aiogram.types.menu_button_commands),
                                                                                                                                                                    198
                                                                                                                                            MenuButtonDefault
MDL (aiogram.enums.currency.Currency attribute), 543
                                                                                                                                                                                                                                 (class
                                                                                                                                                                                                                                                                               in
media\ (aiogram.methods.edit\_message\_media.EditMessageMedia
                                                                                                                                                                   aiogram.types.menu_button_default), 199
                       attribute), 503
                                                                                                                                            MenuButtonType
                                                                                                                                                                                                                            (class
                                                                                                                                                                                                                                                                              in
media(aiogram.methods.send_media_group.SendMediaGroup
                                                                                                                                                                    aiogram.enums.menu_button_type), 547
                      attribute), 449
                                                                                                                                            MenuButtonWebApp
                                                                                                                                                                                                                               (class
                                                                                                                                                                                                                                                                               in
media (aiogram.methods.send_paid_media.SendPaidMedia
                                                                                                                                                                    aiogram.types.menu_button_web_app), 199
                       attribute), 455
                                                                                                                                            MESSAGE (aiogram.enums.update_type.UpdateType at-
media (aiogram.types.input_media_animation.InputMediaAnimation tribute), 552
                       attribute), 184
                                                                                                                                            message (aiogram.handlers.callback_query.CallbackQueryHandler
media(aiogram.types.input_media_audio.InputMediaAudio
                                                                                                                                                                   property), 631
                       attribute), 185
                                                                                                                                            message (aiogram.types.business_intro.BusinessIntro at-
media (aiogram.types.input_media_document.InputMediaDocument tribute), 29
                       attribute), 186
                                                                                                                                            message (aiogram.types.callback_query.CallbackQuery
media(aiogram.types.input_media_photo.InputMediaPhoto
                                                                                                                                                                    attribute), 31
                       attribute), 187
                                                                                                                                            message (aiogram.types.passport_element_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportElement_error_data_field.PassportE
media(aiogram.types.input_media_video.InputMediaVideo
                                                                                                                                                                    attribute), 347
                       attribute), 188
                                                                                                                                            message (aiogram.types.passport_element_error_file.PassportElementErro
media (aiogram.types.input_paid_media_photo.InputPaidMediaPhotoattribute), 347
                       attribute), 189
                                                                                                                                            message (aiogram.types.passport_element_error_files.PassportElementErr
media (aiogram.types.input_paid_media_video.InputPaidMediaVideoattribute), 348
                       attribute), 190
                                                                                                                                            message (aiogram.types.passport_element_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportElement_error_front_side.PassportE
media_group_id (aiogram.types.message.Message at-
                                                                                                                                                                    attribute), 348
                      tribute), 202
                                                                                                                                            message (aiogram.types.passport_element_error_reverse_side.PassportEle
MediaGroupBuilder
                                                                                                                                  in
                                                                                                                                                                    attribute), 349
                       aiogram.utils.media_group), 659
                                                                                                                                            message (aiogram.types.passport_element_error_selfie.PassportElementEn
MEMBER (aiogram.enums.chat_member_status.ChatMemberStatus
                                                                                                                                                                    attribute), 350
                      attribute), 539
                                                                                                                                            message (aiogram.types.passport_element_error_translation_file.Passport_
member_limit(aiogram.methods.create chat invite link.CreateChatuvibaltenk350
                       attribute), 391
                                                                                                                                            message (aiogram.types.passport_element_error_translation_files.Passpor
member_limit(aiogram.methods.edit_chat_invite_link.EditChatInvite_ltinibute), 351
                                                                                                                                            message (aiogram.types.passport_element_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_error_unspecified.PassportElement_
                       attribute), 400
member_limit(aiogram.types.chat invite link.ChatInviteLink
                                                                                                                                                                    attribute), 352
                       attribute), 58
                                                                                                                                            message (aiogram.types.update.Update attribute), 353
                                                                                                                                            Message (class in aiogram.types.message), 199
member_status_changed
                       (aiogram.filters.chat_member_updated.ChatMembressbradgrechtittor_delete_time (aiogram.types.chat.Chat
                       attribute), 573
                                                                                                                                                                    attribute), 36
MemoryStorage (class in aiogram.fsm.storage.memory), message_auto_delete_time
                                                                                                                                                                    (aiogram.types.chat_full_info.ChatFullInfo
MENTION (aiogram.enums.message_entity_type.MessageEntityType
                                                                                                                                                                   attribute), 56
                       attribute), 548
                                                                                                                                            message_auto_delete_time
mention (aiogram.filters.command.CommandObject at-
                                                                                                                                                                    (aiogram.types.message_auto_delete_timer_changed.MessageAut
                       tribute), 572
                                                                                                                                                                    attribute), 255
mention_html() (aiogram.types.user.User method),
                                                                                                                                           MESSAGE_AUTO_DELETE_TIMER_CHANGED
                       271
                                                                                                                                                                    (aiogram.enums.content_type.ContentType
mention_markdown()
                                                                           (aiogram.types.user.User
                                                                                                                                                                    attribute), 540
```

```
message_auto_delete_timer_changed
                                                                                                  attribute), 516
              (aiogram.types.message.Message
                                                                  attribute), message_id(aiogram.methods.pin_chat_message.PinChatMessage
                                                                                                  attribute), 424
message_effect_id(aiogram.methods.send_animation.Samelssaignetjiod(aiogram.methods.set_game_score.SetGameScore
              attribute), 433
                                                                                                  attribute), 519
attribute), 481
              attribute), 436
message_effect_id(aiogram.methods.send_contact.Sendmossage_id(aiogram.methods.stop_message_live_location.StopMessageL
              attribute), 440
                                                                                                  attribute), 508
message_effect_id(aiogram.methods.send_dice.SendDimmessage_id
                                                                                                                 (aiogram.methods.stop_poll.StopPoll
             attribute), 442
                                                                                                  attribute), 509
message_effect_id(aiogram.methods.send_document.SendDsage.ind(aiogram.methods.unpin_chat_message.UnpinChatMessage
             attribute), 445
                                                                                                  attribute), 495
message_effect_id(aiogram.methods.send_game.SendGmessage_id(aiogram.types.external_reply_info.ExternalReplyInfo
                                                                                                  attribute), 132
              attribute), 517
message_effect_id(aiogram.methods.send_invoice.SendMnessiage_id(aiogram.types.inaccessible_message.InaccessibleMessage
             attribute), 530
                                                                                                  attribute), 138
message_effect_id (aiogram.methods.send_location.Senathessation_id (aiogram.types.message.Message attribute),
             attribute), 448
message_effect_id(aiogram.methods.send_media_groupm&sstdyfedixdGnoipgram.types.message_id.MessageId at-
             attribute), 450
                                                                                                  tribute), 256
\verb|message_effect_id| (aiogram.methods.send\_message.Sen \textit{mlstassage}\_id| (aiogram.types.message\_origin\_channel.MessageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.messageOriginChannel.m
                                                                                                  attribute), 257
              attribute), 453
message_effect_id(aiogram.methods.send_photo.SendPhotssage_id(aiogram.types.message_reaction_count_updated.MessageRea
              attribute), 458
                                                                                                  attribute), 259
message_effect_id(aiogram.methods.send_poll.SendPolmessage_id(aiogram.types.message_reaction_updated.MessageReactionU
              attribute), 461
                                                                                                  attribute), 260
message_effect_id(aiogram.methods.send_sticker.SendSnicksage_id(aiogram.types.reply_parameters.ReplyParameters
             attribute), 368
                                                                                                  attribute), 267
message_effect_id(aiogram.methods.send_venue.SendVmessage_ids(aiogram.methods.copy_messages.CopyMessages
              attribute), 464
                                                                                                  attribute), 390
message_effect_id(aiogram.methods.send_video.SendVinlessage_ids(aiogram.methods.delete_messages.DeleteMessages
             attribute), 467
                                                                                                  attribute), 498
message_effect_id(aiogram.methods.send_video_note.Smed\vidge\)Modes (aiogram.methods.forward_messages.ForwardMessages
              attribute), 470
                                                                                                  attribute), 407
message_effect_id(aiogram.methods.send_voice.SendVoicessage_ids(aiogram.types.business_messages_deleted.BusinessMessage
             attribute), 472
                                                                                                  attribute), 30
message_id(aiogram.methods.copy_message.CopyMessagMESSAGE_REACTION(aiogram.enums.update_type.UpdateType
              attribute), 388
                                                                                                  attribute), 552
message_id(aiogram.methods.delete_message.DeleteMessmessage_reaction (aiogram.types.update.Update at-
                                                                                                  tribute), 354
             attribute), 497
{\tt message\_id} \ (aiogram.methods.edit\_message\_caption.Edit {\tt MESSAGE\_apea.6}{\tt TION\_COUNT}) \\
                                                                                                  (aiogram.enums.update_type.UpdateType
             attribute), 499
message_id(aiogram.methods.edit_message_live_location.EditMessage_live_location
              attribute), 501
                                                                                    message_reaction_count
message_id(aiogram.methods.edit_message_media.EditMessageMediaiogram.types.update.Update
                                                                                                                                                       attribute),
             attribute), 503
                                                                                                  354
message_id(aiogram.methods.edit_message_reply_markumEsistMessugaReploMaurkuppes.input_text_message_content.InputTextMessage_id
             attribute), 504
                                                                                                  attribute), 323
```

attribute), 506

attribute), 406

message_id(aiogram.methods.edit_message_text.EditMessage_thread_id(aiogram.methods.close_forum_topic.CloseForumTo

message_id(aiogram.methods.forward_message.ForwardMessage_thread_id(aiogram.methods.copy_message.CopyMessage

message_id(aiogram.methods.get game high scores.Get@messMigh_Shread_id(aiogram.methods.copy messages.CopyMessages

attribute), 385

attribute), 388

attribute), 390 message_thread_id(aiogram.methods.delete_forum_to_	nic DeleteFi	aiogram.types.messag ผ วิจัก Topic	e_auto_delete_timer_	changed),
attribute), 398	Message:	_	(class	in
message_thread_id(aiogram.methods.edit_forum_topid	_	•	`	ııı
attribute), 403		EntityType	(class	in
message_thread_id(aiogram.methods.forward_messag	_		`	in
attribute), 406		Id (class in aiogram.ty		
				_
message_thread_id(aiogram.methods.forward_messag	estres manyer		(class	in
attribute), 407	a w Was P a away	aiogram.types.message		:
message_thread_id(aiogram.methods.reopen_forum_to	opriessayye		(class	in
attribute), 427	G 14 :	aiogram.types.message	e_origin_cnannei),	
message_thread_id(aiogram.methods.send_animation.			. T	
attribute), 432	_	OriginChat	(class	in
${\tt message_thread_id} (\it aiogram.methods.send_audio.S$		aiogram.types.message	-	
attribute), 435	_	OriginHiddenUser	(class	in
${\tt message_thread_id} (\it aiogram.methods.send_chat_actio$	n.SendChat	A xitigr am.types.messag	e_origin_hidden_usei	·),
attribute), 438		258		
<pre>message_thread_id(aiogram.methods.send_contact.Set</pre>	nd Kesisag e	OriginType	(class	in
attribute), 439		aiogram.enums.messa	ge_origin_type), 548	
<pre>message_thread_id(aiogram.methods.send_dice.SendI</pre>	Di M essage	OriginUser	(class	in
attribute), 442		aiogram.types.messago	e_origin_user), 258	
<pre>message_thread_id(aiogram.methods.send_document.</pre>	Se ndDwage i		_	in
attribute), 444	J	aiogram.types.message		dated),
message_thread_id(aiogram.methods.send_game.Send	lGame	259	1	,,
attribute), 517		ReactionUpdated	(class	in
message_thread_id(aiogram.methods.send_invoice.Sen	_	_	,	
attribute), 529	ilairi i o te e	259	_reaction_upatited);	
message_thread_id(aiogram.methods.send_location.Se	en Mh.dc hlt ew		(class	in
attribute), 447		aiogram.dispatcher.mi	`	
message_thread_id(aiogram.methods.send_media_gro	ourM.S.G.R.AMF	-	action condition, 570	
attribute), 450	принания	(aiogram.enums.conte	nt type ContentType	
message_thread_id(aiogram.methods.send_message.Se	ondMossaga		m_iype.comemiype	
attribute), 452	_	_from_chat_id		
	_		a Massaca attrib	(ta)
message_thread_id(aiogram.methods.send_photo.Send	ar noto		ge.Message attribi	ue),
attribute), 457		204		C T
message_thread_id(aiogram.methods.send_poll.SendF	OUTILGRAIE.		n.enums.content_type	.ContentType
attribute), 460	10.51	attribute), 540		
message_thread_id(aiogram.methods.send_sticker.Sen	<i>d Sinucgrate</i> .	_	n.types.message.Mess	rage
attribute), 368		attribute), 204		
${\tt message_thread_id} ({\it aiogram.methods.send_venue.Send})$	<i>dVming</i> erate.		n.types.response_para	imeters.ResponsePai
attribute), 463		attribute), 268		
${\tt message_thread_id} (\it aiogram.methods.send_video.S$	<i>lVinlie</i> me_ty		imation.Animation	at-
attribute), 466		tribute), 20		
<pre>message_thread_id(aiogram.methods.send_video_note</pre>	e.SminNeideyo	pk ot e tiogram.types.audi	o.Audio attribute), 20)
attribute), 469	mime_ty	pe (aiogram.types.de	ocument.Document	at-
<pre>message_thread_id(aiogram.methods.send_voice.Send</pre>	lVoice	tribute), 131		
attribute), 472	mime_ty	pe (aiogram.types.inlin	e_query_result_docu	ment.InlineQueryRe.
<pre>message_thread_id(aiogram.methods.unpin_all_forum</pre>	n_topic_mes	satgeibilda)pihAHForumT	TopicMessages	
attribute), 493		pe (aiogram.types.inlin		.InlineQueryResultV
message_thread_id(aiogram.types.forum_topic.Forum		attribute), 316	v_ ···_	~ ,
attribute), 134	-	pe (aiogram.types.vide	o.Video attribute). 27	4
message_thread_id (aiogram.types.message.Message		pe (aiogram.types.voic		
attribute), 201		gram.enums.currency.C		
	module	,		
MessageAllTolleletellmerthangen iclass in				

```
aiogram.dispatcher.flags, 620
                                               aiogram.methods.copy_message, 387
                                               aiogram.methods.copy_messages, 389
aiogram.enums.bot_command_scope_type, 537
aiogram.enums.chat_action, 538
                                               aiogram.methods.create_chat_invite_link,
aiogram.enums.chat_boost_source_type, 538
aiogram.enums.chat_member_status,539
                                               aiogram.methods.create_chat_subscription_invite_link,
aiogram.enums.chat_type, 539
aiogram.enums.content_type, 539
                                               aiogram.methods.create_forum_topic, 393
aiogram.enums.currency, 541
                                               aiogram.methods.create_invoice_link, 523
aiogram.enums.dice_emoji, 544
                                               aiogram.methods.create_new_sticker_set,
aiogram.enums.encrypted_passport_element,
                                               aiogram.methods.decline_chat_join_request,
aiogram.enums.inline_query_result_type,
                                               aiogram.methods.delete_chat_photo, 396
aiogram.enums.input_media_type, 546
                                               aiogram.methods.delete_chat_sticker_set,
aiogram.enums.input_paid_media_type, 546
aiogram.enums.keyboard_button_poll_type_type,
                                               aiogram.methods.delete_forum_topic, 398
                                               aiogram.methods.delete_message, 496
aiogram.enums.mask_position_point, 547
                                               aiogram.methods.delete_messages, 498
aiogram.enums.menu_button_type, 547
                                               aiogram.methods.delete_my_commands, 399
aiogram.enums.message_entity_type, 548
                                               aiogram.methods.delete_sticker_from_set,
aiogram.enums.message_origin_type, 548
                                                   359
aiogram.enums.paid_media_type, 549
                                               aiogram.methods.delete_sticker_set, 360
                                               aiogram.methods.delete_webhook, 531
aiogram.enums.parse_mode, 549
aiogram.enums.passport_element_error_type,
                                               aiogram.methods.edit_chat_invite_link,
    549
                                                   400
aiogram.enums.poll_type, 550
                                               aiogram.methods.edit_chat_subscription_invite_link,
aiogram.enums.reaction_type_type, 550
aiogram.enums.revenue_withdrawal_state_type,
                                               aiogram.methods.edit_forum_topic, 402
                                               aiogram.methods.edit_general_forum_topic,
aiogram.enums.sticker_format, 551
aiogram.enums.sticker_type, 551
                                               aiogram.methods.edit_message_caption, 499
aiogram.enums.topic_icon_color,551
                                               aiogram.methods.edit_message_live_location,
aiogram.enums.transaction_partner_type,
    552
                                               aiogram.methods.edit_message_media, 503
aiogram.enums.update_type, 552
                                               aiogram.methods.edit_message_reply_markup,
aiogram.exceptions, 618
aiogram.handlers.callback_query, 631
                                               aiogram.methods.edit_message_text, 505
aiogram.methods.add_sticker_to_set, 357
                                               aiogram.methods.edit_user_star_subscription,
aiogram.methods.answer_callback_query,
                                               aiogram.methods.export_chat_invite_link,
aiogram.methods.answer_inline_query, 510
aiogram.methods.answer_pre_checkout_query,
                                               aiogram.methods.forward_message, 406
                                               aiogram.methods.forward_messages, 407
aiogram.methods.answer_shipping_query,
                                               aiogram.methods.get_available_gifts, 361
                                               aiogram.methods.get_business_connection,
aiogram.methods.answer_web_app_query, 512
aiogram.methods.approve_chat_join_request,
                                               aiogram.methods.get_chat, 409
                                               aiogram.methods.get_chat_administrators,
aiogram.methods.ban_chat_member, 382
aiogram.methods.ban_chat_sender_chat, 383
                                               aiogram.methods.get_chat_member, 411
aiogram.methods.close, 384
                                               aiogram.methods.get_chat_member_count,
aiogram.methods.close_forum_topic, 385
aiogram.methods.close_general_forum_topic,
                                               aiogram.methods.get_chat_menu_button, 412
    386
                                               aiogram.methods.get_custom_emoji_stickers,
```

```
362
                                               aiogram.methods.send_paid_media, 454
aiogram.methods.get_file, 413
                                               aiogram.methods.send_photo, 456
aiogram.methods.get_forum_topic_icon_stickers,aiogram.methods.send_poll, 459
                                               aiogram.methods.send_sticker, 367
aiogram.methods.get_game_high_scores, 516
                                               aiogram.methods.send_venue, 463
aiogram.methods.get_me, 415
                                               aiogram.methods.send_video, 465
aiogram.methods.get_my_commands, 415
                                               aiogram.methods.send_video_note, 469
aiogram.methods.get_my_default_administrator_raighgrsam.methods.send_voice, 471
    416
                                               aiogram.methods.set_chat_administrator_custom_title,
aiogram.methods.get_my_description, 417
aiogram.methods.get_my_name, 418
                                               aiogram.methods.set_chat_description, 475
aiogram.methods.get_my_short_description,
                                               aiogram.methods.set_chat_menu_button, 476
                                               aiogram.methods.set_chat_permissions, 477
                                               aiogram.methods.set_chat_photo, 478
aiogram.methods.get_star_transactions,
    526
                                               aiogram.methods.set_chat_sticker_set,479
aiogram.methods.get_sticker_set, 363
                                               aiogram.methods.set_chat_title, 480
aiogram.methods.get_updates, 532
                                               aiogram.methods.set_custom_emoji_sticker_set_thumbnail
aiogram.methods.get_user_chat_boosts,419
aiogram.methods.get_user_profile_photos,
                                               aiogram.methods.set_game_score, 519
                                               aiogram.methods.set_message_reaction, 481
aiogram.methods.get_webhook_info,533
                                               aiogram.methods.set_my_commands, 483
aiogram.methods.hide_general_forum_topic,
                                               aiogram.methods.set_my_default_administrator_rights,
    421
aiogram.methods.leave_chat, 422
                                               aiogram.methods.set_my_description, 485
aiogram.methods.log_out, 423
                                               aiogram.methods.set_my_name, 486
aiogram.methods.pin_chat_message, 424
                                               aiogram.methods.set_my_short_description,
aiogram.methods.promote_chat_member, 425
aiogram.methods.refund_star_payment, 527
                                               aiogram.methods.set_passport_data_errors,
aiogram.methods.remove_chat_verification,
                                               aiogram.methods.set_sticker_emoji_list,
aiogram.methods.remove_user_verification,
    364
                                               aiogram.methods.set_sticker_keywords, 372
aiogram.methods.reopen_forum_topic, 427
                                               aiogram.methods.set_sticker_mask_position,
aiogram.methods.reopen_general_forum_topic,
                                               aiogram.methods.set_sticker_position_in_set,
aiogram.methods.replace_sticker_in_set,
                                               aiogram.methods.set_sticker_set_thumbnail,
aiogram.methods.restrict_chat_member, 429
aiogram.methods.revoke_chat_invite_link,
                                               aiogram.methods.set_sticker_set_title,
aiogram.methods.save_prepared_inline_message, aiogram.methods.set_user_emoji_status,
    513
aiogram.methods.send_animation, 432
                                               aiogram.methods.set_webhook, 534
aiogram.methods.send_audio, 435
                                               aiogram.methods.stop_message_live_location,
aiogram.methods.send_chat_action, 437
aiogram.methods.send_contact, 439
                                               aiogram.methods.stop_poll, 509
aiogram.methods.send_dice, 441
                                               aiogram.methods.unban_chat_member, 489
aiogram.methods.send_document, 443
                                               aiogram.methods.unban_chat_sender_chat,
aiogram.methods.send_game, 517
aiogram.methods.send_gift, 366
                                               aiogram.methods.unhide_general_forum_topic,
aiogram.methods.send_invoice, 528
aiogram.methods.send_location, 446
                                               aiogram.methods.unpin_all_chat_messages,
aiogram.methods.send_media_group, 449
aiogram.methods.send_message, 451
                                               aiogram.methods.unpin_all_forum_topic_messages,
```

```
493
                                               aiogram.types.chat_administrator_rights,
aiogram.methods.unpin_all_general_forum_topic_messages,
                                               aiogram.types.chat_background, 50
aiogram.methods.unpin_chat_message, 495
                                               aiogram.types.chat_boost, 50
aiogram.methods.upload_sticker_file, 377
                                               aiogram.types.chat_boost_added, 50
aiogram.methods.verify_chat, 378
                                               aiogram.types.chat_boost_removed, 51
aiogram.methods.verify_user, 379
                                               aiogram.types.chat_boost_source, 51
                                               aiogram.types.chat_boost_source_gift_code,
aiogram.types.affiliate_info, 325
aiogram.types.animation, 19
                                               aiogram.types.chat_boost_source_giveaway,
aiogram.types.audio, 20
aiogram.types.background_fill, 21
aiogram.types.background_fill_freeform_gradientiogram.types.chat_boost_source_premium,
aiogram.types.background_fill_gradient,
                                               aiogram.types.chat_boost_updated, 53
                                               aiogram.types.chat_full_info,53
aiogram.types.background_fill_solid, 22
                                               aiogram.types.chat_invite_link, 57
aiogram.types.background_type, 22
                                               aiogram.types.chat_join_request, 58
aiogram.types.background_type_chat_theme,
                                               aiogram.types.chat_location, 100
                                               aiogram.types.chat_member, 100
aiogram.types.background_type_fill, 23
                                               aiogram.types.chat_member_administrator,
aiogram.types.background_type_pattern, 23
                                                   101
aiogram.types.background_type_wallpaper,
                                               aiogram.types.chat_member_banned, 102
    24
                                               aiogram.types.chat_member_left, 103
aiogram.types.birthdate, 24
                                               aiogram.types.chat_member_member, 103
                                               aiogram.types.chat_member_owner, 104
aiogram.types.bot_command, 25
aiogram.types.bot_command_scope, 25
                                               aiogram.types.chat_member_restricted, 104
aiogram.types.bot_command_scope_all_chat_adminaisdgnatmortsypes.chat_member_updated, 106
                                               aiogram.types.chat_permissions, 127
aiogram.types.bot_command_scope_all_group_chataiogram.types.chat_photo, 128
                                               aiogram.types.chat_shared, 129
aiogram.types.bot_command_scope_all_private_chantesgram.types.chosen_inline_result, 278
                                               aiogram.types.contact, 129
aiogram.types.bot_command_scope_chat, 26
                                               aiogram.types.copy_text_button, 130
aiogram.types.bot_command_scope_chat_administrantogrsam.types.dice, 130
                                               aiogram.types.document, 130
aiogram.types.bot_command_scope_chat_member,
                                               aiogram.types.encrypted_credentials, 344
                                               aiogram.types.encrypted_passport_element,
aiogram.types.bot_command_scope_default,
                                               aiogram.types.error_event,617
aiogram.types.bot_description, 28
                                               aiogram.types.external_reply_info, 131
aiogram.types.bot_name, 28
                                               aiogram.types.file, 133
aiogram.types.bot_short_description, 28
                                               aiogram.types.force_reply, 133
aiogram.types.business_connection, 29
                                               aiogram.types.forum_topic, 134
aiogram.types.business_intro, 29
                                               aiogram.types.forum_topic_closed, 134
                                               aiogram.types.forum_topic_created, 135
aiogram.types.business_location, 30
aiogram.types.business_messages_deleted,
                                               aiogram.types.forum_topic_edited, 135
                                               aiogram.types.forum_topic_reopened, 135
aiogram.types.business_opening_hours, 30
                                               aiogram.types.game, 356
aiogram.types.business_opening_hours_interval,aiogram.types.game_high_score,357
                                               aiogram.types.general_forum_topic_hidden,
aiogram.types.callback_game, 356
                                               aiogram.types.general_forum_topic_unhidden,
aiogram.types.callback_query, 31
aiogram.types.chat, 32
                                                   136
                                               aiogram.types.gift, 340
```

```
aiogram.types.gifts, 340
                                               aiogram.types.input_file, 182
aiogram.types.giveaway, 136
                                               aiogram.types.input_invoice_message_content,
aiogram.types.giveaway_completed, 137
aiogram.types.giveaway_created, 137
                                               aiogram.types.input_location_message_content,
aiogram.types.giveaway_winners, 137
aiogram.types.inaccessible_message, 138
                                               aiogram.types.input_media, 183
aiogram.types.inline_keyboard_button, 181
                                               aiogram.types.input_media_animation, 183
aiogram.types.inline_keyboard_markup, 182
                                               aiogram.types.input_media_audio, 184
aiogram.types.inline_query, 278
                                               aiogram.types.input_media_document, 186
aiogram.types.inline_query_result, 279
                                               aiogram.types.input_media_photo, 187
aiogram.types.inline_query_result_article,
                                               aiogram.types.input_media_video, 187
                                               aiogram.types.input_message_content, 323
aiogram.types.inline_query_result_audio,
                                               aiogram.types.input_paid_media, 189
                                               aiogram.types.input_paid_media_photo, 189
    282
aiogram.types.inline_query_result_cached_audicaiogram.types.input_paid_media_video, 190
                                               aiogram.types.input_poll_option, 191
aiogram.types.inline_query_result_cached_documarintgram.types.input_sticker, 340
                                               aiogram.types.input_text_message_content,
aiogram.types.inline_query_result_cached_gif,
                                               aiogram.types.input_venue_message_content,
aiogram.types.inline_query_result_cached_mpeg4_gif324
                                               aiogram.types.invoice, 326
aiogram.types.inline_query_result_cached_photoaiogram.types.keyboard_button, 191
                                               aiogram.types.keyboard_button_poll_type,
aiogram.types.inline_query_result_cached_sticker, 192
                                               aiogram.types.keyboard_button_request_chat,
aiogram.types.inline_query_result_cached_video,
                                                   192
                                               aiogram.types.keyboard_button_request_user,
aiogram.types.inline_query_result_cached_voice,
                                               aiogram.types.keyboard_button_request_users,
aiogram.types.inline_query_result_contact,
    300
                                               aiogram.types.labeled_price, 326
aiogram.types.inline_query_result_document,
                                               aiogram.types.link_preview_options, 196
                                               aiogram.types.location, 196
aiogram.types.inline_query_result_game,
                                               aiogram.types.login_url, 197
                                               aiogram.types.mask_position, 341
aiogram.types.inline_query_result_gif,
                                               aiogram.types.maybe_inaccessible_message,
aiogram.types.inline_query_result_location,
                                               aiogram.types.menu_button, 198
                                               aiogram.types.menu_button_commands, 198
aiogram.types.inline_query_result_mpeg4_gif,
                                               aiogram.types.menu_button_default, 199
                                               aiogram.types.menu_button_web_app, 199
aiogram.types.inline_query_result_photo,
                                               aiogram.types.message, 199
                                               aiogram.types.message_auto_delete_timer_changed,
    311
aiogram.types.inline_query_result_venue,
    313
                                               aiogram.types.message_entity, 255
aiogram.types.inline_query_result_video,
                                               aiogram.types.message_id, 256
    314
                                               aiogram.types.message_origin, 256
                                               \verb"aiogram.types.message_origin_channel", 257
aiogram.types.inline_query_result_voice,
                                               aiogram.types.message_origin_chat, 257
aiogram.types.inline_query_results_button,
                                               aiogram.types.message_origin_hidden_user,
aiogram.types.input_contact_message_content,
                                               aiogram.types.message_origin_user, 258
   318
```

```
aiogram.types.message_reaction_count_updated, aiogram.types.revenue_withdrawal_state_failed,
                                                   329
aiogram.types.message_reaction_updated,
                                               aiogram.types.revenue_withdrawal_state_pending,
aiogram.types.order_info, 327
                                               aiogram.types.revenue_withdrawal_state_succeeded,
aiogram.types.paid_media, 260
aiogram.types.paid_media_info, 260
                                               aiogram.types.sent_web_app_message, 325
aiogram.types.paid_media_photo, 261
                                               aiogram.types.shared_user, 268
aiogram.types.paid_media_preview, 261
                                               aiogram.types.shipping_address, 331
                                               aiogram.types.shipping_option, 331
aiogram.types.paid_media_purchased, 327
aiogram.types.paid_media_video, 261
                                               aiogram.types.shipping_query, 332
aiogram.types.passport_data, 345
                                               aiogram.types.star_transaction, 333
aiogram.types.passport_element_error, 346
                                               aiogram.types.star_transactions, 334
aiogram.types.passport_element_error_data_fielatiogram.types.sticker,341
                                               aiogram.types.sticker_set,343
aiogram.types.passport_element_error_file,
                                               aiogram.types.story, 269
                                               aiogram.types.successful_payment, 334
aiogram.types.passport_element_error_files,
                                               aiogram.types.switch_inline_query_chosen_chat,
aiogram.types.passport_element_error_front_sideiogram.types.text_quote, 270
                                               aiogram.types.transaction_partner, 335
aiogram.types.passport_element_error_reverse_sainegram.types.transaction_partner_affiliate_program,
                                                   335
aiogram.types.passport_element_error_selfie,
                                               aiogram.types.transaction_partner_chat,
                                                   336
aiogram.types.passport_element_error_translatianingfildme, types.transaction_partner_fragment,
aiogram.types.passport_element_error_translatiaoiogfidmestypes.transaction_partner_other,
aiogram.types.passport_element_error_unspecifiæribgram.types.transaction_partner_telegram_ads,
aiogram.types.passport_file, 352
                                               aiogram.types.transaction_partner_telegram_api,
aiogram.types.photo_size, 262
aiogram.types.poll, 262
                                               aiogram.types.transaction_partner_user,
aiogram.types.poll_answer, 263
aiogram.types.poll_option, 264
                                               aiogram.types.update, 353
aiogram.types.pre_checkout_query, 327
                                               aiogram.types.user, 270
aiogram.types.prepared_inline_message,
                                               aiogram.types.user_chat_boosts, 272
                                               aiogram.types.user_profile_photos, 272
aiogram.types.proximity_alert_triggered,
                                               aiogram.types.user_shared, 272
                                               aiogram.types.users_shared, 273
aiogram.types.reaction_count, 264
                                               aiogram.types.venue, 273
aiogram.types.reaction_type, 265
                                               aiogram.types.video, 274
                                               aiogram.types.video_chat_ended, 275
aiogram.types.reaction_type_custom_emoji,
                                               aiogram.types.video_chat_participants_invited,
                                                   275
aiogram.types.reaction_type_emoji, 265
aiogram.types.reaction_type_paid, 266
                                               aiogram.types.video_chat_scheduled, 275
                                               aiogram.types.video_chat_started, 275
aiogram.types.refunded_payment, 328
aiogram.types.reply_keyboard_markup, 266
                                               aiogram.types.video_note, 276
aiogram.types.reply_keyboard_remove, 267
                                               aiogram.types.voice, 276
aiogram.types.reply_parameters, 267
                                               aiogram.types.web_app_data,277
aiogram.types.response_parameters, 268
                                               aiogram.types.web_app_info, 277
aiogram.types.revenue_withdrawal_state,
                                               aiogram.types.webhook_info,355
    329
                                               aiogram.types.write_access_allowed, 277
```

```
MongoStorage (class in aiogram.fsm.storage.mongo), name
                                                                   (aiogram.methods.set my name.SetMyName
                                                                 attribute), 486
month (aiogram.types.birthdate.Birthdate attribute), 24
                                                       name (aiogram.methods.set sticker set thumbnail.SetStickerSetThumbnail
MOUTH (aiogram.enums.mask_position_point.MaskPositionPoint
                                                                 attribute), 375
         attribute), 547
                                                       name (aiogram.methods.set_sticker_set_title.SetStickerSetTitle
mpeq4_duration(aiogram.types.inline query result mpeg4 gif.InlinetOibenteResVoltMpeg4Gif
         attribute), 310
                                                       name (aiogram.types.bot name.BotName attribute), 28
mpeg4_file_id(aiogram.types.inline_query_result_cached_ampegdiogrifdInltyp@uchgRanult@dihkdMhpelpdGtifLink at-
         attribute), 291
                                                                 tribute), 57
MPEG4_GIF (aiogram.enums.inline_query_result_type.Inline@mer (Riosghtillytspes.forum_topic.ForumTopic attribute),
         attribute), 545
mpeg4_height (aiogram.types.inline_query_result_mpeg4_mathdline@wwmxlRpsulfMprag46pfc_created.ForumTopicCreated
         attribute), 310
                                                                 attribute), 135
mpeg4_url (aiogram.types.inline_query_result_mpeg4_gif.heline Quiegydrestslpd/splograffiiftopic_edited.ForumTopicEdited
         attribute), 310
                                                                 attribute), 135
mpeg4_width (aiogram.types.inline_query_result_mpeg4_mifatedi(reiQevarmNexpelsMpleg4Gifo.OrderInfo attribute), 327
         attribute), 310
                                                       name (aiogram.types.sticker_set.StickerSet attribute), 343
MUR (aiogram.enums.currency.Currency attribute), 543
                                                       nanostar_amount(aiogram.types.affiliate info.AffiliateInfo
                                                                 attribute), 326
MVR (aiogram.enums.currency.Currency attribute), 543
MXN (aiogram.enums.currency.Currency attribute), 543
                                                       nanostar_amount (aiogram.types.star transaction.StarTransaction
MY_CHAT_MEMBER (aiogram.enums.update_type.UpdateType
                                                                 attribute), 333
         attribute), 553
                                                       need_email(aiogram.methods.create_invoice_link.CreateInvoiceLink
my_chat_member
                   (aiogram.types.update.Update
                                                                 attribute), 524
         tribute), 354
                                                       need_email(aiogram.methods.send invoice.SendInvoice
MYR (aiogram.enums.currency.Currency attribute), 543
                                                                 attribute), 530
MZN (aiogram.enums.currency.Currency attribute), 543
                                                       need_email(aiogram.types.input_invoice_message_content.InputInvoiceM
                                                                 attribute), 321
Ν
                                                       need_name (aiogram.methods.create_invoice_link.CreateInvoiceLink
                                                                 attribute), 524
name (aiogram.methods.add_sticker_to_set.AddStickerToSet
                                                       need_name (aiogram.methods.send_invoice.SendInvoice
         attribute), 357
{\tt name} \ (aiogram.methods.create\_chat\_invite\_link.CreateChatInviteLink} {\tt attribute}), 530
                                                       need_name(aiogram.types.input_invoice_message_content.InputInvoiceMe
         attribute), 391
name (aiogram.methods.create_chat_subscription_invite_link.CreateCHLtishuserishionInviteLink
                                                       need_phone_number (aiogram.methods.create_invoice_link.CreateInvoice
         attribute), 393
                                                                 attribute), 524
name (aiogram.methods.create_forum_topic.CreateForumTopic
                                                       need_phone_number(aiogram.methods.send invoice.SendInvoice
         attribute), 394
\verb|name| (a iogram.methods.create\_new\_sticker\_set.CreateNewStickerSet| attribute), 530
                                                       need_phone_number (aiogram.types.input_invoice_message_content.Inpu
         attribute), 358
                                                                 attribute), 321
name (aiogram.methods.delete_sticker_set.DeleteStickerSet
                                                       need_shipping_address
         attribute), 360
                                                                 (aiogram.methods.create invoice link.CreateInvoiceLink
name (aiogram.methods.edit_chat_invite_link.EditChatInviteLink
                                                                 attribute), 524
         attribute), 400
name (aiogram.methods.edit_chat_subscription_invite_link.Panchahinsing)tashingseLink
                                                                 (aiogram.methods.send_invoice.SendInvoice
         attribute), 402
                                                                 attribute), 530
name (aiogram.methods.edit forum topic.EditForumTopic
                                                       need_shipping_address
         attribute), 403
attribute), 322
         attribute), 404
                                                       needs_repainting(aiogram.methods.create_new_sticker_set.CreateNews
name (aiogram.methods.get_sticker_set.GetStickerSet at-
                                                                 attribute), 359
         tribute), 363
\verb|name|| (a iogram.methods.replace\_sticker\_in\_set.ReplaceSticker_ing)|
                                                                                 (aiogram.types.sticker.Sticker
                                                                 attribute), 342
         attribute), 365
name (aiogram.methods.set_custom_emoji_sticker_set_thumbawiissatcusememojissatkersepphahatamember_updated.ChatMemberUp
                                                                 attribute), 106
         attribute), 370
```

NEW_CHA	AT_MEMBERS (aiogram.enums.content_type.Conten			.business_opening_ho	urs.BusinessOpening
nou cha	attribute), 540 at_members (aiogram.types.message.Message		attribute), 30	as business enemine b	ours internal Pusine
	attribute), 203		attribute), 31		
NEW_CHA	AT_PHOTO (aiogram.enums.content_type.ContentTy attribute), 540	_	ids (aiogram.types.po tribute), 263	ll_answer.PollAnswer	at-
new_cha	at_photo (aiogram.types.message.Message at- tribute), 203	_	(aiogram.methods.stribute), 460	send_poll.SendPoll	at-
NEW_CHA	T_TITLE (aiogram.enums.content_type.ContentTy attribute), 540		(aiogram.types.poll.Pa nfo(aiogram.types.pre		CheckoutQuery
new_cha	at_title (aiogram.types.message.Message at- tribute), 203		attribute), 328 nfo (aiogram.types.suc		
new_rea	action (aiogram.types.message_reaction_updated attribute), 260	.MessageR			•
next_of	Efset (aiogram.methods.answer_inline_query.Ansattribute), 511	w eerloglin e@			eplyInfo
NGN (aio	gram.enums.currency.Currency attribute), 543		ogram.enums.transact	ion_partner_type.Tran	sactionPartnerType
	gram.enums.currency.Currency attribute), 543 gram.enums.currency.Currency attribute), 543		attribute), 552		
	gram.enums.currency.Currency attribute), 543	Р			
	gram.enums.currency.Currency attribute), 543	PAB (aiog	ram.enums.currency.C	urrency attribute), 543	3
0		pack()	-	lback_data.CallbackD	
offset((aiogram.methods.get_star_transactions.GetStarTr attribute), 526		gram.enums.reaction_ attribute), 550	type_type.ReactionTyp	еТуре
offset	(aiogram.methods.get_updates.GetUpdates attribute), 532		DIA (aiogram.enums.c attribute), 539	content_type.ContentTy	vpe
offset(aiogram.methods.get_user_profile_photos.GetUse attribute), 420		dias (aiogram.types.ext attribute), 132	ernal_reply_info.Exter	rnalReplyInfo
offset	(aiogram.types.inline_query.InlineQuery at-tribute), 278	paid_med	dia (<i>aiogram.types.me</i> 202	ssage.Message attribu	te),
offset((aiogram.types.message_entity.MessageEntity attribute), 256		dia (aiogram.types.pai attribute), 260	d_media_info.PaidMe	diaInfo
ok (aiogi	ram.methods.answer_pre_checkout_query.Answer. attribute), 520		น่า<u>ผู</u>นล่าง gram.types.tra attribute), 339	nsaction_partner_user	:TransactionPartner0
ok (aiogi	ram.methods.answer_shipping_query.AnswerShipp attribute), 521		dia_payload(aiogran attribute), 327	n.types.paid_media_pi	ırchased.PaidMediaI
old_cha	at_member (aiogram.types.chat_member_updated. attribute), 106	(PaidMenel		n.types.transaction_pa	rtner_user.Transactio
old_rea	action(aiogram.types.message_reaction_updated	.MaidMed	ie a c tilonsUpdaiec gram.ty	pes.paid_media), 260	
	attribute), 260	PaidMed:		(class	in
old_sti	.cker (aiogram.methods.replace_sticker_in_set.Re	-		•	
	attribute), 365	PaidMed:		(class	in
	ne_keyboard(aiogram.types.reply_keyboard_manattribute), 266	PaidMed:	iaPreview	(class	in
only_if	E_banned (aiogram.methods.unban_chat_member.			_	•
only no	attribute), 489 w_members (aiogram.types.giveaway.Giveaway		iaPurchased aiogram.types.paid_m	(class	in
OILLY_ILE	attribute), 136	PaidMed:		(class	in
only ne	w_members (aiogram.types.giveaway_winners.Gi			•	
	attribute), 138	PaidMed:		(class	in
open_pe	eriod (aiogram.methods.send_poll.SendPoll at-		aiogram.types.paid_m	*	
- •	tribute), 461		ode (<i>aiogram.client.de</i>)		rties
open_pe	eriod (aiogram.types.poll.Poll attribute), 263		attribute), 557	•	

```
parse_mode(aiogram.methods.copy_message.CopyMessagparse_mode(aiogram.types.input_media_document.InputMediaDocument
                  attribute), 388
                                                                                                                              attribute), 186
parse_mode(aiogram.methods.edit message caption.Edit.Meassage@antogram.types.input media photo.InputMediaPhoto
                                                                                                                              attribute), 187
                 attribute), 499
parse_mode (aiogram.methods.edit_message_text.EditMesspar Eextmode (aiogram.types.input_media_video.InputMediaVideo
                 attribute), 506
                                                                                                                              attribute), 188
parse_mode(aiogram.methods.send animation.SendAnimapianse_mode(aiogram.types.input text message content.InputTextMessage
                 attribute), 433
                                                                                                                              attribute), 324
parse_mode (aiogram.methods.send_audio.SendAudio parse_webapp_init_data()
                                                                                                                                                                                   (in
                                                                                                                                                                                                      module
                                                                                                                              aiogram.utils.web_app), 646
                 attribute), 435
parse_mode (aiogram.methods.send_document.SendDocumRearseMode (class in aiogram.enums.parse_mode), 549
                                                                                                           PASSPORT (aiogram.enums.encrypted_passport_element.EncryptedPasspor
                 attribute), 445
parse_mode(aiogram.methods.send_message.SendMessage
                                                                                                                              attribute), 545
                 attribute), 452
                                                                                                           PASSPORT_DATA (aiogram.enums.content_type.ContentType
{\tt parse\_mode} \ (aiogram.methods.send\_paid\_media.SendPaidMedia
                                                                                                                             attribute), 541
                 attribute), 455
                                                                                                           passport_data
                                                                                                                                                        (aiogram.types.message.Message
                          (aiogram.methods.send_photo.SendPhoto
                                                                                                                              attribute), 205
parse_mode
                                                                                                           PASSPORT_REGISTRATION
                 attribute), 457
parse_mode
                           (aiogram.methods.send_video.SendVideo
                                                                                                                              (aiogram.enums.encrypted_passport_element.EncryptedPassport
                                                                                                                              attribute), 545
                 attribute), 467
parse_mode
                             (aiogram.methods.send_voice.SendVoice PassportData (class in aiogram.types.passport_data),
                 attribute), 472
parse_mode (aiogram.types.inline_query_result_audio.InliReQsportAsulmentarror
                                                                                                                                                                                (class
                                                                                                                                                                                                                in
                 attribute), 283
                                                                                                                              aiogram.types.passport element error),
parse_mode (aiogram.types.inline_query_result_cached_audio.Inline@toeryResultCachedAudio
                 attribute), 285
                                                                                                           PassportElementErrorDataField
parse_mode(aiogram.types.inline_query_result_cached_document.Inling@www.tyResupluCxpubuad_Plomenenerror_data_field),
                 attribute), 287
parse_mode (aiogram.types.inline_query_result_cached_gift.inline@utEhentiEnchedGife
                                                                                                                                                                                     (class
                                                                                                                                                                                                                in
                 attribute), 289
                                                                                                                              aiogram.types.passport_element_error_file),
parse_mode(aiogram.types.inline_query_result_cached_mpeg4_gif.InlineQueryResultCachedMpeg4Gif
                 attribute), 291
                                                                                                           PassportElementErrorFiles
                                                                                                                                                                                      (class
                                                                                                                                                                                                                in
parse\_mode (a iogram.types.in line\_query\_result\_cached\_photo. In line \textit{QagryRhetsyle} \textit{C.pcdsepBho}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{topclsepBho}}\underline{\textit{t
                 attribute), 294
parse_mode(aiogram.types.inline_query_result_cached_villes.spbirat@uemeRetEduConFredWit&icde
                 attribute), 298
                                                                                                                              aiogram.types.passport element error front side),
parse_mode (aiogram.types.inline query result cached voice.Inline@ueryResultCachedVoice
                  attribute), 300
                                                                                                           PassportElementErrorReverseSide
                                                                                                                                                                                             (class
parse_mode (aiogram.types.inline_query_result_document.InlineQueridResult.Duperspenstport_element_error_reverse_side),
                                                                                                                              349
                 attribute), 303
parse_mode(aiogram.types.inline_query_result_gif.Inline@nessphornEGefmentErrorSelfie
                                                                                                                                                                                       (class
                                                                                                                                                                                                                in
                                                                                                                              aiogram.types.passport_element_error_selfie),
                 attribute), 305
parse_mode (aiogram.types.inline_query_result_mpeg4_gif.InlineQu@A\ResultMpeg4Gif
                 attribute), 310
                                                                                                           PassportElementErrorTranslationFile (class in
parse_mode (aiogram.types.inline_query_result_photo.InlineQueryResialtRhostsypes.passport_element_error_translation_file),
                 attribute), 312
parse_mode(aiogram.types.inline_query_result_video.Inlinea@ssporReElletMedetErrorTranslationFiles (class in
                 attribute), 316
                                                                                                                             aiogram.types.passport_element_error_translation_files),
parse_mode(aiogram.types.inline_query_result_voice.InlineQueryResultVoice
                                                                                                           PassportElementErrorType
                 attribute), 317
                                                                                                                                                                                     (class
                                                                                                                                                                                                                in
parse_mode (aiogram.types.input_media_animation.InputMediaAnimatigram.enums.passport_element_error_type),
                 attribute), 184
                                                                                                                              549
parse_mode(aiogram.types.input media audio.InputMedia AustiportElementErrorUnspecified
                                                                                                                                                                                             (class
                 attribute), 185
                                                                                                                              aiogram.types.passport element error unspecified),
```

351		attribute), 345
PassportFile (class in aiogram.types.passport_file), 352	phone_	<pre>number (aiogram.types.inline_query_result_contact.InlineQueryRe attribute), 301</pre>
pattern (aiogram.filters.exception.ExceptionMessageFilte attribute), 581	rphone_	<pre>number (aiogram.types.input_contact_message_content.InputConta attribute), 318</pre>
pay (aiogram.types.inline_keyboard_button.InlineKeyboar attribute), 182	·d Bhane_	_number (aiogram.types.order_info.OrderInfo at- tribute), 327
pay_for_upgrade (aiogram.methods.send_gift.SendGift attribute), 366		<pre>Jumber (class in aiogram.utils.formatting), 657 (aiogram.enums.content_type.ContentType at-</pre>
payload (aiogram.methods.create_invoice_link.CreateInvoice_link), 523	oiceLink	
payload (aiogram.methods.send_invoice.SendInvoice at- tribute), 529		attribute), 545 (aiogram.enums.input_media_type.InputMediaType
payload (aiogram.methods.send_paid_media.SendPaidMeattribute), 455	edia	attribute), 546 (aiogram.enums.input_paid_media_type.InputPaidMediaType
payload (aiogram.types.input_invoice_message_content.In		
attribute), 321		(aiogram.enums.paid_media_type.PaidMediaType
PEN (aiogram.enums.currency.Currency attribute), 543	1110101	attribute), 549
PENDING (aiogram.enums.revenue_withdrawal_state_type.	Pahata	
attribute), 550	Приклик	tribute), 457
pending_join_request_count	photo	(aiogram.methods.set_chat_photo.SetChatPhoto
(aiogram.types.chat_invite_link.ChatInviteLink	photo	attribute), 478
attribute), 58	nhoto	(aiogram.types.chat.Chat attribute), 36
pending_update_count	photo	
(aiogram.types.webhook_info.WebhookInfo	photo	tribute), 55
attribute), 355	photo	(aiogram.types.chat_shared.ChatShared at-
performer (aiogram.methods.send_audio.SendAudio at-	Piloto	tribute), 129
<i>tribute</i>), 436	photo((aiogram.types.external_reply_info.ExternalReplyInfo
performer (aiogram.types.audio.Audio attribute), 20	_	attribute), 132
<pre>performer(aiogram.types.inline_query_result_audio.Inli</pre>	n ep@ae.ro yi	ReingltAmultypes.game.Game attribute), 356
attribute), 283	photo	(aiogram.types.message.Message attribute), 202
performer (aiogram.types.input_media_audio.InputMedia_attribute), 185		attribute), 261
permissions (aiogram.methods.restrict_chat_member.Re attribute), 429	s tphø€b a	atM @nibgr am.types.shared_user.SharedUser at- tribute), 269
permissions (aiogram.methods.set_chat_permissions.Set attribute), 477	CphaoReo 1	afii lien i d (aiogram.types.inline_query_result_cached_photo.Inline (attribute), 294
permissions (aiogram.types.chat.Chat attribute), 36 permissions (aiogram.types.chat_full_info.ChatFullInfo	photo_	height (aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 524
attribute), 56	photo_	height (aiogram.methods.send_invoice.SendInvoice
<pre>personal_chat (aiogram.types.chat.Chat attribute), 36</pre>		attribute), 529
<pre>personal_chat (aiogram.types.chat_full_info.ChatFullIng</pre>	<i>fa</i> photo_	<pre>height (aiogram.types.inline_query_result_photo.InlineQueryResu attribute), 312</pre>
PERSONAL_DETAILS (aiogram.enums.encrypted_passport_attribute), 545	_epkrokot.	Enight(dPosspon:tPleminu t_invoice_message_content.InputInvoic attribute), 321
	е рЬБ іло <u>г</u>	Spitzad (aus spoon fi. herendout ds. create_invoice_link. Create Invoice Link attribute), 524
PHONE_NUMBER (aiogram.enums.message_entity_type.Mes. attribute), 548	sapladinoii	
	a ø hoto	_size(aiogram.types.input_invoice_message_content.InputInvoiceN
attribute), 439	1	attribute), 321
	photo_	_url (aiogram.methods.create_invoice_link.CreateInvoiceLink
tribute), 129		attribute), 524
nhone number (giogram types encrypted passport eleme	nn Fatou	ntrad Phairmont Edonath to de sand invoice Sand Invoice

attribute), 529 photo_url (aiogram.types.inline_query_result_photo.Inl	PollType (class in aiogram.enums.poll_type), 550 postatyRxx(util)gotum.methods.set_sticker_position_in_set.SetStickerPositio
attribute), 311	attribute), 374
photo_url (aiogram.types.input_invoice_message_content attribute), 321	t. popsitfixmi&eMogrsangaGnest.gau me_high_score.GameHighScore attribute), 357
photo_url (aiogram.utils.web_app.WebAppChat at- tribute), 649	position (aiogram.types.text_quote.TextQuote at- tribute), 270
photo_url (aiogram.utils.web_app.WebAppUser at- tribute), 648	<pre>post_code (aiogram.types.shipping_address.ShippingAddress</pre>
	eHRE(icidgirdm.enums.message_entity_type.MessageEntityType attribute), 548
photo_width (aiogram.methods.send_invoice.SendInvoice attribute), 529	Pre (class in aiogram.utils.formatting), 657 PRE_CHECKOUT_QUERY (aiogram.enums.update_type.UpdateType
photo_width(aiogram.types.inline_query_result_photo.In	
attribute), 312	<pre>pre_checkout_query (aiogram.types.update.Update at-</pre>
<pre>photo_width(aiogram.types.input_invoice_message_cont</pre>	
attribute), 321	pre_checkout_query_id
photos (aiogram.types.user_profile_photos.UserProfilePhotos.UserPr	attribute), 520
PhotoSize (class in aiogram.types.photo_size), 262	PreCheckoutQuery (class in
PHP (aiogram.enums.currency.Currency attribute), 543	aiogram.types.pre_checkout_query), 327
pin() (aiogram.types.message.Message method), 252	prefer_large_media(aiogram.types.link_preview_options.LinkPreview
pin_message() (aiogram.types.chat.Chat method), 43	attribute), 196
	prefer_small_media(aiogram.types.link_preview_options.LinkPreview
aiogram.methods.pin_chat_message), 424	attribute), 196
PINNED_MESSAGE (aiogram.enums.content_type.ContentTy_attribute), 540	tribute), 572
pinned_message (aiogram.types.chat.Chat attribute), 36	PREMIUM (aiogram.enums.chat_boost_source_type.ChatBoostSourceType attribute), 538
$\verb pinned_message (a iogram. types. chat_full_info. ChatFull Info. ChatFull Inf$	
attribute), 56	tribute), 342
pinned_message (aiogram.types.message.Message at- tribute), 204	(aiogram.types.giveaway.Giveaway attribute),
PKR (aiogram.enums.currency.Currency attribute), 543	136
PLN (aiogram.enums.currency.Currency attribute), 543	premium_subscription_month_count
point (aiogram.types.mask_position.MaskPosition attribute), 341	(aiogram.types.giveaway_winners.GiveawayWinners attribute), 138
POLL (aiogram.enums.content_type.ContentType at- tribute), 540	prepare_value() (aiogram.client.session.base.BaseSession method), 16
POLL (aiogram.enums.update_type.UpdateType at-tribute), 553	PreparedInlineMessage (class in aiogram.types.prepared_inline_message),
poll (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 133	PREVIEW (aiogram.enums.paid_media_type.PaidMediaType
poll (aiogram.types.message.Message attribute), 203	attribute), 549
poll (aiogram.types.update.Update attribute), 354	<pre>prices (aiogram.methods.create_invoice_link.CreateInvoiceLink</pre>
Poll (class in aiogram.types.poll), 262	attribute), 523
POLL_ANSWER (aiogram.enums.update_type.UpdateType attribute), 553	prices (aiogram.methods.send_invoice.SendInvoice attribute), 529
poll_answer (aiogram.types.update.Update attribute), 354	<pre>prices (aiogram.types.input_invoice_message_content.InputInvoiceMess</pre>
poll_id (aiogram.types.poll_answer.PollAnswer at- tribute), 263	<pre>prices (aiogram.types.shipping_option.ShippingOption</pre>
PollAnswer (class in aiogram.types.poll_answer), 263 PollOption (class in aiogram.types.poll_option), 264	PRIVATE (aiogram.enums.chat_type.ChatType attribute), 539

```
prize_description(aiogram.types.giveaway.Giveaway protect_content(aiogram.methods.send_message.SendMessage
                   attribute), 136
                                                                                                                                    attribute), 453
prize_description (aiogram.types.giveaway_winners.GipanotayWinnantent (aiogram.methods.send_paid_media.SendPaidMedia
                                                                                                                                    attribute), 455
                   attribute), 138
prize_star_count (aiogram.types.chat_boost_source_givparous_CluaBnostSoluciogfivecumeshods.send_photo.SendPhoto
                  attribute), 52
                                                                                                                                    attribute), 458
prize_star_count (aiogram.types.giveaway.Giveaway protect_content(aiogram.methods.send_poll.SendPoll
                                                                                                                                    attribute), 461
                   attribute), 136
prize_star_count (aiogram.types.giveaway_created.Givepromy@atateont(aiogram.methods.send_sticker.SendSticker
                                                                                                                                    attribute), 368
                  attribute), 137
\verb|prize_star_count| (a iogram. types. give away\_winners. Giv| \verb|prize_star_count| (a iogram. methods. send\_venue. Send Venue on the first of the f
                  attribute), 138
                                                                                                                                    attribute), 464
profile_accent_color_id (aiogram.types.chat.Chat
                                                                                                                protect_content (aiogram.methods.send_video.SendVideo
                  attribute), 36
                                                                                                                                    attribute), 467
profile_accent_color_id
                                                                                                                 {\tt protect\_content} \ (a iogram.methods.send\_video\_note.SendVideoNote
                  (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                                    attribute), 470
                  attribute), 55
                                                                                                                 protect_content(aiogram.methods.send_voice.SendVoice
profile_background_custom_emoji_id
                                                                                                                                    attribute), 472
                   (aiogram.types.chat.Chat attribute), 37
                                                                                                                 provider_data(aiogram.methods.create_invoice_link.CreateInvoiceLink
profile_background_custom_emoji_id
                                                                                                                                    attribute), 524
                  (aiogram.types.chat_full_info.ChatFullInfo
                                                                                                                 provider_data(aiogram.methods.send_invoice.SendInvoice
                  attribute), 55
                                                                                                                                    attribute), 529
promote() (aiogram.types.chat.Chat method), 44
                                                                                                                 provider_data(aiogram.types.input_invoice_message_content.InputInvoi
PromoteChatMember
                                                                                                                                    attribute), 321
                                                                   (class
                                                                                                        in
                  aiogram.methods.promote_chat_member),
                                                                                                                 provider_payment_charge_id
                                                                                                                                    (aiogram.types.refunded_payment.RefundedPayment
{\tt protect\_content} \ (a iogram. client. default. DefaultBotProperties
                                                                                                                                    attribute), 329
                                                                                                                 provider_payment_charge_id
                  attribute), 557
protect_content(aiogram.methods.copy_message.CopyMessage
                                                                                                                                   (aiogram.types.successful_payment.SuccessfulPayment
                  attribute), 388
                                                                                                                                    attribute), 334
protect_content (aiogram.methods.copy_messages.Copythosxiglesr_token (aiogram.methods.create_invoice_link.CreateInvoiceLink
                  attribute), 390
                                                                                                                                    attribute), 523
protect_content (aiogram.methods.forward_message.Fopproviiders.ageken (aiogram.methods.send_invoice.SendInvoice
                                                                                                                                    attribute), 529
                  attribute), 406
protect_content (aiogram.methods.forward_messages.Fapnowil@ess.tip&esn (aiogram.types.input_invoice_message_content.InputInv.
                  attribute), 408
                                                                                                                                    attribute), 321
protect_content(aiogram.methods.send_animation.Sendphroizaintioty_alert_radius
                   attribute), 433
                                                                                                                                    (aiogram.methods.edit_message_live_location.EditMessageLiveL
protect\_content (aiogram.methods.send\_audio.SendAudio
                                                                                                                                    attribute), 502
                                                                                                                 proximity_alert_radius
                  attribute), 436
protect_content(aiogram.methods.send contact.SendContact
                                                                                                                                    (aiogram.methods.send location.SendLocation
                  attribute), 440
                                                                                                                                    attribute), 447
protect_content(aiogram.methods.send_dice.SendDice proximity_alert_radius
                                                                                                                                    (aiogram.types.inline_query_result_location.InlineQueryResultLo
                  attribute), 442
protect_content(aiogram.methods.send_document.SendDocumentattribute), 307
                                                                                                                 proximity_alert_radius
                  attribute), 445
protect_content(aiogram.methods.send_game.SendGame
                                                                                                                                    (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), 517
                                                                                                                                    attribute), 323
protect_content(aiogram.methods.send_invoice.SendInvoiceximity_alert_radius
                                                                                                                                    (aiogram.types.location.Location
                  attribute), 530
                                                                                                                                                                                                           attribute),
{\tt protect\_content} \ (a iogram.methods.send\_location.SendLocation
                                                                                                                 PROXIMITY_ALERT_TRIGGERED
                  attribute), 448
protect_content (aiogram.methods.send_media_group.SendMedia Gniovepam.enums.content_type.ContentType
                                                                                                                                    attribute), 541
                  attribute), 450
```

<pre>proximity_alert_triggered (aiogram.types.message.Message attribute),</pre>	${\tt reaction} \ (aiogram.methods.set_message_reaction.SetMessageReaction \\ attribute), 482$
205	ReactionCount (class in
ProximityAlertTriggered (class in	aiogram.types.reaction_count), 264
<pre>aiogram.types.proximity_alert_triggered), 264</pre>	reactions (aiogram.types.message_reaction_count_updated.MessageReaction), 259
PURCHASED_PAID_MEDIA (aiogram.enums.update_type.UpdateType	ReactionType (class in aiogram.types.reaction_type), 265
attribute), 553	ReactionTypeCustomEmoji (class in
purchased_paid_media (aiogram.types.update.Update attribute), 354	<pre>aiogram.types.reaction_type_custom_emoji), 265</pre>
PYG (aiogram.enums.currency.Currency attribute), 543	ReactionTypeEmoji (class in
Python Enhancement Proposals	aiogram.types.reaction_type_emoji), 265
PEP 484, 3	ReactionTypePaid (class in
PEP 492, 3	aiogram.types.reaction_type_paid), 266
	ReactionTypeType (class in
Q	aiogram.enums.reaction_type_type), 550
QAR (aiogram.enums.currency.Currency attribute), 543 query (aiogram.types.chosen_inline_result.ChosenInlineRe	read() (aiogram.types.input_file.BufferedInputFile method), 182
attribute), 278	read() (aiogram.types.input_file.FSInputFile method),
query (aiogram.types.inline_query.InlineQuery at- tribute), 278	183 read() (aiogram.types.input_file.InputFile method), 182
query (aiogram.types.switch_inline_query_chosen_chat.Sv attribute), 269	
query_id (aiogram.utils.web_app.WebAppInitData at- tribute), 647	<pre>receiver(aiogram.types.star_transaction.StarTransaction</pre>
question (aiogram.methods.send_poll.SendPoll at- tribute), 460	receiver (aiogram.utils.web_app.WebAppInitData at- tribute), 647
question (aiogram.types.poll.Poll attribute), 262	RECORD_VIDEO (aiogram.enums.chat_action.ChatAction
question_entities (aiogram.methods.send_poll.SendPo attribute), 460	
question_entities (aiogram.types.poll.Poll attribute),	class method), 644
263	RECORD_VIDEO_NOTE (aiogram.enums.chat_action.ChatAction attribute), 538
question_parse_mode (aiogram.methods.send_poll.SendPoll at-	record_video_note()
tribute), 460	(aiogram.utils.chat_action.ChatActionSender
QUIZ (aiogram.enums.keyboard_button_poll_type_type.Key attribute), 547	RECORD_VOICE (aiogram.enums.chat_action.ChatAction
QUIZ (aiogram.enums.poll_type.PollType attribute), 550	attribute), 538
quote (aiogram.types.message.Message attribute), 201	record_voice() (aiogram.utils.chat_action.ChatActionSender
quote (aiogram.types.reply_parameters.ReplyParameters	class method), 644
attribute), 267	RED (aiogram.enums.topic_icon_color.TopicIconColor
quote_entities (aiogram.types.reply_parameters.ReplyF	Parameters attribute), 551
	RedisStorage (class in aiogram.fsm.storage.redis), 592 [REGUNDED SPAYMENT (aiogram.enums.content_type.ContentType
attribute), 268	attribute), 540 pagafynded_payment (aiogram.types.message.Message
${\tt quote_position} \ (a iogram. types. reply_parameters. Reply Fattribute), 268$	attribute), 204
R	RefundedPayment (class in aiogram.types.refunded_payment), 328
raw_state(aiogram.dispatcher.middlewares.data.Middlewares	RefundStarPayment (class in
attribute), 570	aiogram.methods.refund_star_payment), 527
react() (aiogram.types.message.Message method), 253	regexp_match(aiogram.filters.command.CommandObject

attribute), 572	method), 209
register() (aiogram.fsm.scene.SceneRegistry method),	${\tt reply_contact()}\ (a iogram. types. in accessible_message. In accessible Message. The accessible message is a contact of the accessible message in accessible message. The accessible message is a contact of the accessible message is a contact of the accessible message. The accessible message is a contact of the accessible message is a contact of the accessible message is a contact of the accessible message. The accessible message is a contact of the accessible message is a cont$
610	method), 147
<pre>register() (aiogram.webhook.aiohttp_server.BaseReque</pre>	es il-phole contact() (aiogram.types.message.Message method), 212
<pre>register() (aiogram.webhook.aiohttp_server.SimpleRequent</pre>	uww#Mmddeice() (aiogram.types.inaccessible_message.InaccessibleMessage method), 166
register() (aiogram.webhook.aiohttp_server.TokenBased method), 623	
REGULAR (aiogram.enums.keyboard_button_poll_type_type	e. KephyardtBumærPo(NypioTypio n.types.inaccessible_message.InaccessibleMe.
attribute), 547	method), 149
REGULAR (aiogram.enums.poll_type.PollType attribute), 550	method), 213
REGULAR (aiogram.enums.sticker_type.StickerType attribute), 551	reply_game() (aiogram.types.inaccessible_message.InaccessibleMessage method), 151
remaining_count (aiogram.types.gift.Gift attribute), 340	reply_game() (aiogram.types.message.Message method), 216
remove_caption (aiogram.methods.copy_messages.Copy.attribute), 390	Massinvoice() (aiogram.types.inaccessible_message.InaccessibleMess method), 154
remove_date(aiogram.types.chat_boost_removed.ChatBoattribute), 51	
<i>"</i>	e. NeplyKdydoardNn(n) (aiogram.types.inaccessible_message.InaccessibleMe. method), 157
	reply_location() (aiogram.types.message.Message
$aiogram.methods.remove_chat_verification),$	method), 221
363 REMOVED_CHAT_BOOST (aiogram.enums.update_type.Upda	reply_markup(aiogram.methods.copy_message.CopyMessage attType attribute), 388
<pre>attribute), 553 removed_chat_boost(aiogram.types.update.Update at-</pre>	reply_markup (aiogram.methods.edit_message_caption.EditMessageCapt attribute), 499
tribute), 355	reply_markup (aiogram.methods.edit_message_live_location.EditMessage
RemoveUserVerification (class in	attribute), 502
aiogram.methods.remove_user_verification), 364	reply_markup(aiogram.methods.edit_message_media.EditMessageMedia attribute), 503
render() (aiogram.utils.formatting.Text method), 655	reply_markup(aiogram.methods.edit_message_reply_markup.EditMessage
RENTAL_AGREEMENT (aiogram.enums.encrypted_passport_	_element.En attyjhuteP q5&p6ortElement
attribute), 545	$\verb"reply_markup" (a iogram.methods.edit_message_text.EditMessageText$
ReopenForumTopic (class in	attribute), 506
aiogram.methods.reopen_forum_topic), 427	reply_markup(aiogram.methods.send_animation.SendAnimation
ReopenGeneralForumTopic (class in	attribute), 433
atogram.metnoas.reopen_generat_jorum_topic), 428	reply_markup(aiogram.methods.send_audio.SendAudio attribute), 436
ReplaceStickerInSet (class in	reply_markup(aiogram.methods.send_contact.SendContact
aiogram.methods.replace_sticker_in_set), 365	attribute), 440 reply_markup (aiogram.methods.send_dice.SendDice
reply() (aiogram.types.inaccessible_message.Inaccessibl	
method), 140	reply_markup(aiogram.methods.send_document.SendDocument
reply() (aiogram.types.message.Message method), 225	attribute), 445
	ge rlepplyeswihlkMfe(xityg eram.methods.send_game.SendGame
method), 142	attribute), 517
- · · · · · · · · · · · · · · · · · · ·	$\verb"reply_markup" (a iogram. methods. send_invoice. Send Invoice$
method), 206	attribute), 530
	creplylenteskupe(aiogram.methods.send_location.SendLocation
method), 145	attribute), 448
reply_audio() (aiogram.types.message.Message	reply_markup(aiogram.methods.send_message.SendMessage

```
attribute), 453
                                                                                                                                    attribute), 312
reply_markup(aiogram.methods.send_paid_media.SendPaidMedimarkup(aiogram.types.inline_query_result_venue.InlineQueryResu
                  attribute), 455
                                                                                                                                    attribute), 314
reply_markup(aiogram.methods.send_photo.SendPhoto reply_markup(aiogram.types.inline_query_result_video.InlineQueryResu
                  attribute), 458
                                                                                                                                     attribute), 316
reply_markup(aiogram.methods.send poll.SendPoll at- reply_markup(aiogram.types.inline query result voice.InlineOueryResu
                                                                                                                                    attribute), 318
                  tribute), 461
reply_markup (aiogram.methods.send_sticker.SendSticker reply_markup (aiogram.types.message.Message at-
                  attribute), 369
                                                                                                                                     tribute), 206
reply_markup(aiogram.methods.send_venue.SendVenue reply_media_group()
                  attribute), 464
                                                                                                                                    (aiogram.types.inaccessible_message.InaccessibleMessage
reply_markup(aiogram.methods.send_video.SendVideo
                                                                                                                                    method), 159
                  attribute), 467
                                                                                                                 reply_media_group()
reply_markup(aiogram.methods.send_video_note.SendVideoNote
                                                                                                                                   (aiogram.types.message.Message
                                                                                                                                                                                                             method),
                  attribute), 470
                                                                                                                                     224
reply_markup (aiogram.methods.send_voice.SendVoice reply_paid_media() (aiogram.types.inaccessible_message.InaccessibleN
                  attribute), 472
                                                                                                                                    method), 179
reply_markup(aiogram.methods.stop_message_live_locatioepplreppleisdamedivek)c(atiogram.types.message.Message
                                                                                                                                    method), 254
                  attribute), 508
reply_markup (aiogram.methods.stop_poll.StopPoll at- reply_parameters (aiogram.methods.copy_message.CopyMessage
                  tribute), 509
                                                                                                                                     attribute), 388
reply_markup (aiogram.types.inline_query_result_article.lndpleQpay&naxleAss(cdicogram.methods.send_animation.SendAnimation
                                                                                                                                     attribute), 433
                   attribute), 281
reply_markup(aiogram.types.inline_query_result_audio.InlineQupanRemethAnsl(aiogram.methods.send_audio.SendAudio
                  attribute), 283
                                                                                                                                    attribute), 436
{\tt reply\_markup} (a iogram.types.in line\_query\_result\_cached\_{\tt nepliy\_lphina@erevrR} (suit @mohed & indits.send\_contact.Send Contact) \\
                  attribute), 285
                                                                                                                                    attribute), 440
reply_markup(aiogram.types.inline_query_result_cached_ndmplynaputdekiet@csetylResultaCondthdDosemule_ndice.SendDice
                  attribute), 287
                                                                                                                                     attribute), 442
reply_markup(aiogram.types.inline_query_result_cached_peif_lylipa@aweylexsakt@ioghadCrifethods.send_document.SendDocument
                  attribute), 289
                                                                                                                                     attribute), 445
reply_markup (aiogram.types.inline_query_result_cached_reply_4 prif.hmlirte@sdxijRgsahnGnachadMpend_cipline.SendGame
                  attribute), 291
                                                                                                                                     attribute), 517
reply_markup(aiogram.types.inline_query_result_cached_pdpky_Ipkine@erexRe(ailt@wwhedPhotds.send_invoice.SendInvoice
                  attribute), 294
                                                                                                                                     attribute), 530
reply_markup(aiogram.types.inline_query_result_cached_reightyr_plinenQtery_Residg@awchand&hiodlessend_location.SendLocation
                  attribute), 296
                                                                                                                                     attribute), 448
reply_markup (aiogram.types.inline_query_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cached_neiplay.lphinrequery_result_cac
                   attribute), 298
                                                                                                                                     attribute), 450
reply_markup(aiogram.types.inline_query_result_cached_nepides_Iplina@ete_nResult@cached_nebidioels.send_message.SendMessage
                                                                                                                                    attribute), 453
                  attribute), 300
{\tt reply\_markup} \ (a iogram. types. in line\_query\_result\_contact {\tt MephyQparpRexidtS}) \ (a iogram. types. types. in line\_query\_result\_contact {\tt MephyQparpRexidtS}) \ (a iogram. types. types. types. in line\_query\_result\_contact {\tt MephyQparpRexidtS}) \ (a iogram. types. types.
                  attribute), 301
                                                                                                                                     attribute), 455
reply_markup (aiogram.types.inline_query_result_documenephlynp@nemyttesnstiQnonymen.tmethods.send_photo.SendPhoto
                  attribute), 303
                                                                                                                                    attribute), 458
reply_markup(aiogram.types.inline_query_result_game.Indiape\DucparRamah\Grap(aiogram.methods.send_poll.SendPoll
                  attribute), 303
                                                                                                                                    attribute), 461
reply_markup(aiogram.types.inline_query_result_gif.Inline@pley.yRaraln@tfers(aiogram.methods.send_sticker.SendSticker
                  attribute), 305
                                                                                                                                     attribute), 368
reply_markup(aiogram.types.inline_query_result_location:bpline@aanakevents.orinjmam.methods.send_venue.SendVenue
                  attribute), 307
                                                                                                                                     attribute), 464
reply_markup(aiogram.types.inline_query_result_mpeg4_replyinpa@nemyRersultMpgg466ifnethods.send_video.SendVideo
                  attribute), 310
                                                                                                                                    attribute), 467
reply_markup(aiogram.types.inline query result photo.IndepelQupearRanatiethes(aiogram.methods.send video note.SendVideoNote
```

attribute), 470		(aiogram.methods.send_pa	oll.SendPoll	at-
reply_parameters(aiogram.methods.send_voice.SendVo		<i>tribute</i>), 462		
attribute), 472		o_message_id		
reply_photo() (aiogram.types.inaccessible_message.Ina		_	icker.SendSticker	
method), 161		attribute), 369		
reply_photo() (aiogram.types.message.Message				
method), 227		(aiogram.methods.send_ve	nue.SendVenue	
reply_poll() (aiogram.types.inaccessible_message.Inacc				
method), 164		o_message_id		
reply_poll() (aiogram.types.message.Message		(aiogram.methods.send_vi	deo.SendVideo	
method), 229		attribute), 468		
<pre>reply_sticker() (aiogram.types.inaccessible_message.I</pre>	nnepass <u>ib</u> de	<i>M</i> message_id		
method), 168		(aiogram.methods.send_vi	deo_note.SendVid	leoNote
<pre>reply_sticker() (aiogram.types.message.Message</pre>		attribute), 470		
method), 234	reply_to	o_message_id		
reply_to_message (aiogram.types.message.Message		(aiogram.methods.send_va	oice.SendVoice	
attribute), 201		attribute), 473		
reply_to_message_id	reply_to	o_story (aiogram.types.n	nessage.Message	at-
(aiogram.methods.copy_message.CopyMessage		tribute), 202		
attribute), 388	reply_ve	enue() (aiogram.types.ina	ccessible_messag	e.InaccessibleMessa
reply_to_message_id		method), 170		
$(a iogram. methods. send_animation. Send Animation and A$	nreply_ve	enue() (aiogram.ty _l	pes.message.Mess	age
attribute), 433		method), 235		
reply_to_message_id	reply_v	ideo() (aiogram.types.ina	ccessible_message	e.InaccessibleMessa
$(aiogram.methods.send_audio.SendAudio$		method), 172		
attribute), 436	reply_v	ideo() (aiogram.ty _l	pes.message.Mess	age
reply_to_message_id		method), 237		
$(a iogram.methods.send_contact.SendContact$		ideo_note() (<i>aiogram.typ</i>	es.inaccessible_n	nessage.Inaccessible
attribute), 440		method), 175		
reply_to_message_id	reply_v	ideo_note() (<i>aiogram.ty</i>	pes.message.Mess	age
(aiogram.methods.send_dice.SendDice at-		method), 240		
tribute), 442		oice()(aiogram.types.ina	ccessible_messago	e.InaccessibleMesso
reply_to_message_id		method), 177		
(aiogram.methods.send_document.SendDocument			pes.message.Mess	age
attribute), 445		method), 243		
reply_to_message_id		yboardBuilder	(class	in
(aiogram.methods.send_game.SendGame		aiogram.utils.keyboard), 6		
attribute), 518		yboardMarkup	(class	in
reply_to_message_id		aiogram.types.reply_keybo	oard_markup),	
(aiogram.methods.send_invoice.SendInvoice		266		
attribute), 530		yboardRemove	(class	in
reply_to_message_id		aiogram.types.reply_keybo		1
(aiogram.methods.send_location.SendLocation	ReplyPa		elass	in
attribute), 448		aiogram.types.reply_parar		I.D.
reply_to_message_id		_chat (aiogram.types.keyb	oard_button.Keyb	oardButton
(aiogram.methods.send_media_group.SendMedia	_	attribute), 191		
attribute), 450		_contact (aiogram.types.k	keyboard_button. k	KeyboardButton
reply_to_message_id		attribute), 191		
(aiogram.methods.send_message.SendMessage		_count (aiogram.types.trai	nsaction_partner_	tetegram_api.Trans
attribute), 453		attribute), 338	l 1 Cl Cl	4
reply_to_message_id	_	_id (aiogram.types.chat_sl	iarea.CnatShared	at-
(aiogram.methods.send_photo.SendPhoto		tribute), 129	nd button mani-	t ahat Vanhaandb
attribute), 458		_id(aiogram.types.keyboa	ra_vunon_request	_спаі. к еуроағаВиі
reply_to_message_id		attribute), 193		

```
\verb"request_id" (a iogram. types. keyboard\_button\_request\_user. Keyboard \verb"ButtonRequestUser") and the property of the propert
                   attribute), 194
                                                                                                                  RESTRICTED (aiogram.enums.chat member status.ChatMemberStatus
request_id(aiogram.types.keyboard button request users.KeyboardBritattRequ0stUsers
                                                                                                                  result(aiogram.methods.answer_web_app_query.AnswerWebAppQuery
                   attribute), 195
request_id (aiogram.types.user_shared.UserShared at-
                                                                                                                                      attribute), 512
                  tribute), 272
                                                                                                                  result (aiogram.methods.save prepared inline message.SavePreparedInl
request_id (aiogram.types.users shared.UsersShared
                                                                                                                                      attribute), 515
                                                                                                                   result\_id(aiogram.types.chosen\_inline\_result.ChosenInlineResult
                   attribute), 273
request_location(aiogram.types.keyboard_button.KeyboardButtomattribute), 278
                                                                                                                  \verb"results" (a iogram.methods. answer\_in line\_query. Answer In line Query
                   attribute), 191
request_name(aiogram.types.keyboard_button_request_users.KeyboattdiButto)nReQuestUsers
                                                                                                                   retake() (aiogram.fsm.scene.SceneWizard method), 613
                   attribute), 195
request_photo (aiogram.types.keyboard_button_request_datatyteydbbaedBaitagrReqtspetChastponse_parameters.ResponseParameters
                  attribute), 194
                                                                                                                                      attribute), 268
{\tt request\_photo} \ (a iogram. types. keyboard\_button\_request\_{\tt iReeventingWoothdBratten} \\ {\tt iReeventingWoothdBratten} 
                                                                                                                                                                                              (class
                                                                                                                                                                                                                             in
                  attribute), 195
                                                                                                                                      aiogram.types.revenue_withdrawal_state),
request_poll (aiogram.types.keyboard_button.KeyboardButton
                                                                                                                                      329
                                                                                                                  RevenueWithdrawalStateFailed
                   attribute), 192
                                                                                                                                                                                                      (class
                                                                                                                                                                                                                             in
request_title(aiogram.types.keyboard_button_request_chat.KeybooiogButtanpResquest@thettwithdrawal_state_failed),
                   attribute), 194
request_user(aiogram.types.keyboard_button.KeyboardBateenueWithdrawalStatePending
                                                                                                                                                                                                       (class
                                                                                                                                                                                                                             in
                   attribute), 192
                                                                                                                                      aiogram.types.revenue_withdrawal_state_pending),
request_username(aiogram.types.keyboard_button_request_chat.KeyboardButtonRequestChat
                   attribute), 194
                                                                                                                  RevenueWithdrawalStateSucceeded
request_username(aiogram.types.keyboard_button_request_users.KieybrandlBpttsnnRuquust_Wistndrawal_state_succeeded),
                   attribute), 195
request_users (aiogram.types.keyboard_button.Keyboard\@buttenue\@withdrawalStateType
                                                                                                                                                                                                   (class
                                                                                                                                                                                                                             in
                  attribute), 191
                                                                                                                                      aiogram.enums.revenue_withdrawal_state_type),
request_write_access
                   (aiogram.types.login_url.LoginUrl
                                                                                                                 REVERSE_SIDE (aiogram.enums.passport_element_error_type.PassportEle.
                                                                                          attribute),
                   197
                                                                                                                                      attribute), 549
reset_data_on_enter
                                                                                                                  reverse_side(aiogram.types.encrypted_passport_element.EncryptedPass
                   (aiogram.fsm.scene.SceneConfig
                                                                                                                                      attribute), 345
                                                                                           attribute),
                                                                                                                  revoke_invite_link()
                                                                                                                                                                                 (aiogram.types.chat.Chat
reset_history_on_enter
                                                                                                                                      method), 39
                   (aiogram.fsm.scene.SceneConfig
                                                                                           attribute),
                                                                                                                  revoke_messages (aiogram.methods.ban_chat_member.BanChatMember
                                                                                                                                      attribute), 382
resize_keyboard(aiogram.types.reply_keyboard_markupRecyd)kK@platdIndMittelLipnk
                                                                                                                                                                                            (class
                   attribute), 266
                                                                                                                                      aiogram.methods.revoke_chat_invite_link),
resolve_bot() (aiogram.webhook.aiohttp_server.BaseRequestHandlen0
                   method), 622
                                                                                                                  rights (aiogram.methods.set my default administrator rights.SetMyDefa
resolve_bot() (aiogram.webhook.aiohttp server.SimpleRequestHandleibute), 484
                   method), 623
                                                                                                                  RON (aiogram.enums.currency.Currency attribute), 543
resolve_bot() (aiogram.webhook.aiohttp_server.TokenBcBOBRequeiststandenums.topic_icon_color.TopicIconColor
                   method), 623
                                                                                                                                      attribute), 551
resolve_used_update_types()
                                                                                                                   rotation_angle(aiogram.types.background_fill_gradient.BackgroundFill
                   (aiogram.dispatcher.router.Router
                                                                                             method),
                                                                                                                                      attribute), 22
                   560
                                                                                                                  Router (class in aiogram.dispatcher.router), 559
ResponseParameters
                                                                      (class
                                                                                                          in
                                                                                                                  row()
                                                                                                                                     (aiogram.utils.keyboard.InlineKeyboardBuilder
                  aiogram.types.response_parameters), 268
                                                                                                                                      method), 637
RestartingTelegram, 618
                                                                                                                                      (aiogram.utils.keyboard.ReplyKeyboardBuilder\\
                                                                                                                  row()
restrict() (aiogram.types.chat.Chat method), 45
                                                                                                                                      method), 639
RestrictChatMember
                                                                      (class
                                                                                                          in RSD (aiogram.enums.currency.Currency attribute), 543
                                                                                                                  RUB (aiogram.enums.currency.Currency attribute), 544
                   aiogram.methods.restrict_chat_member),
```

${\it cun_polling()}$ (a $iogram.dispatcher.dispatcher.Dispatcher.$	ersend_phone_number_to_provider
method), 566	(aiogram.methods.create_invoice_link.CreateInvoiceLink attribute), 524
S	send_phone_number_to_provider
<pre>safe_parse_webapp_init_data() (in module</pre>	(aiogram.methods.send_invoice.SendInvoice attribute), 530
SAR (aiogram.enums.currency.Currency attribute), 544	send_phone_number_to_provider
SavePreparedInlineMessage (class in aiogram.methods.save_prepared_inline_message	(aiogram.types.input_invoice_message_content.InputInvoiceMess attribute), 322
513	SendAnimation (class in
scale (aiogram.types.mask_position.MaskPosition at-	aiogram.methods.send_animation), 432
tribute), 341	SendAudio (class in aiogram.methods.send_audio), 435
Scene (class in aiogram.fsm.scene), 609	SendChatAction (class in
SceneConfig (class in aiogram.fsm.scene), 611	aiogram.methods.send_chat_action), 437
SceneException, 618	SendContact (class in aiogram.methods.send_contact),
SceneRegistry (class in aiogram.fsm.scene), 609	439
ScenesManager (class in aiogram.fsm.scene), 610	SendDice (class in aiogram.methods.send_dice), 441
SceneWizard (class in aiogram.fsm.scene), 611	SendDocument (class in
scope (aiogram.methods.delete_my_commands.DeleteMyC attribute), 399	SENDER (aiogram.enums.chat_type.ChatType attribute),
$scope\ (aiogram.methods.get_my_commands.GetMyCommands)$	ands 539
attribute), 415	sender_boost_count (aiogram.types.message.Message
$scope\ (aiogram.methods.set_my_commands.SetMyCommands)$	ands attribute), 201
attribute), 483	sender_business_bot
score (aiogram.methods.set_game_score.SetGameScore attribute), 519	(aiogram.types.message.Message attribute), 201
score (aiogram.types.game_high_score.GameHighScore attribute), 357	sender_chat (aiogram.types.message.Message at- tribute), 201
secret (aiogram.types.encrypted_credentials.EncryptedC	resendars_chat (aiogram.types.message_origin_chat.MessageOriginChat
attribute). 344	attribute), 257
secret_token (aiogram.methods.set_webhook.SetWebhook attribute), 535	<pre>grender_chat_id(aiogram.methods.ban_chat_sender_chat.BanChatSende attribute), 383</pre>
SEK (aiogram.enums.currency.Currency attribute), 544	sender_chat_id(aiogram.methods.unban_chat_sender_chat.UnbanChatS
selective (aiogram.types.force_reply.ForceReply	attribute), 490
attribute), 134	sender_user(aiogram.types.message_origin_user.MessageOriginUser
selective(aiogram.types.reply_keyboard_markup.Reply.	
attribute), 266	sender_user_name (aiogram.types.message_origin_hidden_user.Message
selective(aiogram.types.reply_keyboard_remove.ReplyI	
attribute), 267	SendGame (class in aiogram.methods.send_game), 517
SELFIE (aiogram.enums.passport_element_error_type.Pas	
attribute), 549	SendInvoice (class in aiogram.methods.send_invoice),
selfie(aiogram.types.encrypted_passport_element.Encry	
attribute), 345	SendLocation (class in
send_copy() (aiogram.types.message.Message method),	aiogram.methods.send_location), 446
245	SendMediaGroup (class in
send_email_to_provider	aiogram.methods.send_media_group), 449
(aiogram methods create invoice link CreateInv	osendMessage (class in aiogram.methods.send_message),
attribute), 524	451
send_email_to_provider	SendPaidMedia (class in
(aiogram.methods.send_invoice.SendInvoice	aiogram.methods.send_paid_media), 454
attribute), 530	SendPhoto (class in aiogram.methods.send_photo), 456
send_email_to_provider	SendPol1 (class in aiogram.methods.send_poll), 459
(aiooram tynes innut invoice message content I	กอิลทรี่ใช้เอโหลรัฐใสอเรากับอยู่เอยูram.methods.send_sticker),
attribute), 322	367

SendVenue (class in aiogram. SendVideo (class in aiogram. SendVideoNote aiogram.methods.sen	methods.send_video), 40 (class		aiogram.methods.set_my_commands), 483 SetMyDefaultAdministratorRights (class in aiogram.methods.set_my_default_administrator_rights), 484
SendVoice (class in aiogram.		71	SetMyDescription (class in
SentWebAppMessage	(class	in	aiogram.methods.set_my_description), 485
	web_app_message), 325		SetMyName (class in aiogram.methods.set_my_name),
set_administrator_custon			486
(aiogram.types.chat.			SetMyShortDescription (class in
set_data() (aiogram.fsm.sc.		(be	aiogram.methods.set_my_short_description),
613			487
	n.storage.base.BaseStor	age	SetPassportDataErrors (class in
method), 594			aiogram.methods.set_passport_data_errors),
<pre>set_description() (aiogram</pre>	m.types.chat.Chat metho	(d),	536
46			SetStickerEmojiList (class in
set_locale() (aiogram.utils. method), 641	.i18n.middleware.FSMI	18nN	Aiddleware aiogram.methods.set_sticker_emoji_list), 371
<pre>set_name (aiogram.types.stick)</pre>	ker.Sticker attribute), 34	2	SetStickerKeywords (class in
set_permissions()(aiogram	m.types.chat.Chat metho	(d),	aiogram.methods.set_sticker_keywords), 372
<pre>set_photo() (aiogram.types.</pre>	chat Chat method) 47		SetStickerMaskPosition (class in
set_position_in_set() (and		kor	aiogram.methods.set_sticker_mask_position),
method), 342	ogram.iypes.siieker.siie	KC1	373
**	n.storage.base.BaseStora	ana	SetStickerPositionInSet (class in
method), 594	i.siorage.base.basesiora	uge	aiogram.methods.set_sticker_position_in_set),
set_sticker_set() (aiogram	m types chat Chat methe	(be	374
41	n.types.chai.Chai meino	π),	SetStickerSetThumbnail (class in
<pre>set_title() (aiogram.types.</pre>	chat Chat method) 47		aiogram.methods.set_sticker_set_thumbnail),
SetChatAdministratorCus		in	375
	*		n_SighteStickerSetTitle (class in
474	_cnai_aaminisiraior_ca	isiom	aiogram.methods.set_sticker_set_title), 376
SetChatDescription	(class	in	
=	t_chat_description), 475		method), 642
SetChatMenuButton	class (class		
	t_chat_menu_button),	ııı	aiogram.methods.set_user_emoji_status),
476			488
SetChatPermissions	(class	in	SetWebhook (class in aiogram.methods.set_webhook),
aiogram.methods.set	_chat_permissions),		534
477			SGD (aiogram.enums.currency.Currency attribute), 544
SetChatPhoto	(class	in	SharedUser (class in aiogram.types.shared_user), 268
aiogram.methods.set	* '		shifted_id (aiogram.types.chat.Chat property), 37
SetChatStickerSet	(class	in	shipping_address(aiogram.types.order_info.OrderInfo
	t_chat_sticker_set), 479		attribute), 327
SetChatTitle	(class	in	shipping_address(aiogram.types.shipping_query.ShippingQuery
aiogram.methods.set			attribute), 332
SetCustomEmojiStickerSe		in	$\verb shipping_option_id (a iogram. types.pre_checkout_query. PreCheckoutQ) $
_	_custom_emoji_sticker_	_set_1	thumbnail)attribute), 328
370			shipping_option_id(aiogram.types.successful_payment.SuccessfulPaym
SetGameScore	(class	in	attribute), 334
aiogram.methods.set			shipping_options (aiogram.methods.answer_shipping_query.AnswerShi
SetMessageReaction	(class	in	attribute), 521
aiogram.methods.set 481	t_message_reaction),		SHIPPING_QUERY (aiogram.enums.update_type.UpdateType attribute), 552
SetMyCommands	(class	in	shipping_query (aiogram.types.update.Update at-

tribute), î				attribute), 305
	_	nethods.answer_ship _l	ping_	_ <i>sphow_AraptaiShipphingQuee</i> dia
attribute)				(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResul
ShippingAddress		(class	in	attribute), 310
	types.shipping	_address), 331		show_caption_above_media
ShippingOption aiogram.	types.shipping	(class _option), 331	in	(aiogram.types.inline_query_result_photo.InlineQueryResultPhoattribute), 312
ShippingQuery		(class	in	show_caption_above_media
	types.shipping			(aiogram.types.inline_query_result_video.InlineQueryResultVide
	_	nethods.set_my_shor	t_des	scription.SeaMyiShoe)Delscription
attribute)	*	7 . 7 . 7		show_caption_above_media
attribute)	, 28		_	on.BotShort Daisgriphian pes.input_media_animation.InputMediaAnimation attribute), 184
		es.link_preview_optic	ons.L	Li shBw_lvcapOpom<u>n</u>a bove_media
attribute)				(aiogram.types.input_media_photo.InputMediaPhoto
		nswer_callback_que	ry.An	nswerCallb aatsChete), 187
attribute)	*			show_caption_above_media
snow_alert(alogation property)		ack_answer.Callback	Ansv	wer (aiogram.types.input_media_video.InputMediaVideo attribute), 189
show_caption_ak				show_caption_above_media
		DefaultBotPropertie.	c	(aiogram.types.message.Message attribute),
attribute)		DejamiBon roperne.	,	203
show_caption_ak				SimpleI18nMiddleware (class in
(aiogram	.methods.copy	_message.CopyMess	age	aiogram.utils.i18n.middleware), 640
attribute)	, 388			SimpleRequestHandler (class in
show_caption_ak				aiogram.webhook.aiohttp_server), 622
		_message_caption.Ed	itMes	es SigeCMACH INE (aiogram.enums.dice_emoji.DiceEmoji
attribute)				attribute), 544
show_caption_ak				SLOT_MACHINE (aiogram.types.dice.DiceEmoji at-
		_animation.SendAnir	natio	
<pre>attribute) show_caption_ab</pre>				slow_mode_delay (aiogram.types.chat.Chat attribute), 37
		naid media SendPa	idMa	edsiluow_mode_delay(aiogram.types.chat_full_info.ChatFullInfo
attribute)		_риш_теши.5епа1 а	iuivic	attribute), 56
show_caption_ak				small_file_id (aiogram.types.chat_photo.ChatPhoto
		_photo.SendPhoto		attribute), 128
attribute)		_1		small_file_unique_id
show_caption_ab	ove_media			(aiogram.types.chat_photo.ChatPhoto at-
(aiogram	.methods.send	_video.SendVideo		tribute), 128
attribute)				source (aiogram.types.chat_boost.ChatBoost attribute),
show_caption_ab				50
(aiogram attribute)	• • • • • •	query_result_cached_	_gif.Iı	In BoxQue(yRegultiCity)wexlCri fit_boost_removed.ChatBoostRemoved attribute), 51
show_caption_ab	ove_media			source (aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGift_
(aiogram	.types.inline_q	query_result_cached_	mpeg	g4_gif.Inlin a@ilewyB e5ultCachedMpeg4Gif
attribute)				$\verb source (a iogram.types.chat_boost_source_give away.ChatBoostSourceGiven away.ChatBoostSour$
show_caption_ab			_	attribute), 52
		query_result_cached_	phot	to skolimeQuaivgRasult)GasluhdiP_lboo ost_source_premium.ChatBoostSourcePren
attribute)				attribute), 52
show_caption_ak		mam, nose.141. 1	,,; J.	source (aiogram.types.passport_element_error_data_field.PassportEleme
(aiogram attribute)	• • • • • •	_{[uery_result_cacned_}	_viae	o.InlineQu enyRbsutè)Ca&h edVideo source(aiogram.types.passport_element_error_file.PassportElementErro
show cantion at				source (alogram.types.passport_etement_error_jue.rassportEtementError_

 $(a iogram. types. in line_query_result_gif. In line \textit{Quers_offessalt} \textbf{(} \textit{Gif} ogram. types. passport_element_error_files. PassportElementError_files. PassportEleme$

attribute), 331

```
attribute), 348
                                                                                                                      STATIC
                                                                                                                                            (aiogram.enums.sticker_format.StickerFormat
source (aiogram.types.passport_element_error_front_side.PassportElemintTe); 6tEFrontSide
                   attribute), 348
                                                                                                                      status (aiogram.types.chat_member_administrator.ChatMemberAdministr
source (aiogram.types.passport_element_error_reverse_side.PassportEkihum)EtftdrReverseSide
                   attribute), 349
                                                                                                                      status(aiogram.types.chat\_member\_banned.ChatMemberBanned
\verb|source|| (a iogram.types.passport\_element\_error\_selfie.PassportElement \textit{EribatS} \texttt{plfi0} 2
                   attribute), 350
                                                                                                                      status(aiogram.types.chat_member_left.ChatMemberLeft
Source (aiogram.types.passport_element_error_translation_file.Passpoomibleten);ntErrorTranslationFile
                   attribute), 350
                                                                                                                      status(aiogram.types.chat\_member\_member.ChatMemberMember
source (aiogram.types.passport_element_error_translation_files.PassptnilElten_eMErrorTranslationFiles
                   attribute), 351
                                                                                                                      status(aiogram.types.chat\_member\_owner.ChatMemberOwner)
source (aiogram.types.passport_element_error_unspecified.PassportEtenibuttErri@Unspecified
                                                                                                                      status(aiogram.types.chat_member_restricted.ChatMemberRestricted
                   attribute), 352
source (aiogram.types.star_transaction.StarTransaction
                                                                                                                                          attribute), 104
                   attribute), 333
                                                                                                                      STICKER (aiogram.enums.content_type.ContentType at-
SPOILER (aiogram.enums.message_entity_type.MessageEntityType tribute), 540
                                                                                                                      STICKER (aiogram.enums.inline_query_result_type.InlineQueryResultType
                   attribute), 548
Spoiler (class in aiogram.utils.formatting), 657
                                                                                                                                          attribute), 545
attribute), 336
                                                                                                                                          attribute), 357
attribute), 454
                                                                                                                                          attribute), 359
star_count (aiogram.types.gift.Gift attribute), 340
                                                                                                                      sticker(aiogram.methods.replace\_sticker\_in\_set.ReplaceStickerInSet
star_count(aiogram.types.paid_media_info.PaidMediaInfo
                                                                                                                                          attribute), 365
                                                                                                                      sticker (aiogram.methods.send_sticker.SendSticker at-
                   attribute), 260
start_date(aiogram.types.video_chat_scheduled.VideoChatSchedulterilbute), 368
                   attribute), 275
                                                                                                                      \verb+sticker+ (a iogram.methods.set\_sticker\_emoji\_list.SetStickerEmojiList
start_param (aiogram.utils.web_app.WebAppInitData
                                                                                                                                          attribute), 371
                   attribute), 647
                                                                                                                      sticker(aiogram.methods.set_sticker_keywords.SetStickerKeywords
start_parameter(aiogram.methods.send_invoice.SendInvoice
                                                                                                                                          attribute), 372
                   attribute), 529
                                                                                                                      sticker(aiogram.methods.set_sticker_mask_position.SetStickerMaskPosit
start_parameter(aiogram.types.inline_query_results_button.InlineQtrebyRe)sullsButton
                   attribute), 318
                                                                                                                      sticker(aiogram.methods.set_sticker_position_in_set.SetStickerPositionIn
start_parameter
                                                    (aiogram.types.invoice.Invoice
                                                                                                                                          attribute), 374
                   attribute), 326
                                                                                                                      sticker(aiogram.methods.upload_sticker_file.UploadStickerFile
start_polling() (aiogram.dispatcher.dispatcher.Dispatcher
                                                                                                                                          attribute), 377
                   method), 566
                                                                                                                      sticker (aiogram.types.business_intro.BusinessIntro at-
start\_timestamp(aiogram.methods.send\_video.SendVideo)
                                                                                                                                          tribute), 29
                    attribute), 467
                                                                                                                       \verb+sticker+ (a iogram. types. external\_reply\_info. ExternalReplyInfo
start_timestamp(aiogram.types.input_media_video.InputMediaVidetoribute), 132
                                                                                                                      sticker (aiogram.types.gift.Gift attribute), 340
                   attribute), 188
start_timestamp(aiogram.types.input_paid_media_videoshipak@widMedia@viia@gam.types.input_sticker.InputSticker
                                                                                                                                          attribute), 340
                   attribute), 190
                                                                                                                   sticker (aiogram.types.message.Message attribute), 202
start_timestamp
                                              (aiogram.types.video.Video
                                                                                                                      Sticker (class in aiogram.types.sticker), 341
                   tribute), 274
StarTransaction
                                                                    (class
                                                                                                             in sticker_file_id(aiogram.types.inline_query_result_cached_sticker.Inli
                   aiogram.types.star\_transaction), 333
                                                                                                                                          attribute), 296
                                                                                                                    sticker_format(aiogram.methods.create_new_sticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewSticker_set.CreateNewStic
StarTransactions
                   aiogram.types.star_transactions), 334
                                                                                                                                          attribute), 359
\verb+state+ (aiogram. dispatcher. middle wares. data. Middle ware \textit{Destai}+ \texttt{cker\_format}+ (aiogram. methods. upload\_sticker\_file. UploadStickerFile) + \texttt{cker\_format}+ \texttt{cher\_format}+ \texttt{che
                   attribute), 570
                                                                                                                                          attribute), 377
state (aiogram.fsm.scene.SceneConfig attribute), 611
                                                                                                                      sticker_set_name(aiogram.methods.set_chat_sticker_set.SetChatSticker
{\tt state}\,(aiogram.types.shipping\_address.ShippingAddress
                                                                                                                                          attribute), 479
```

780 Index

sticker_set_name (aiogram.types.chat.Chat attribute),

	${\tt ubscription_price} \ (a iogram. methods. create_chat_subscription_invite_chat_subscription_$
sticker_set_name(aiogram.types.chat_full_info.ChatFullI	
	ubscription_price(aiogram.types.chat_invite_link.ChatInviteLink
sticker_type (aiogram.methods.create_new_sticker_set.Cr attribute), 359	eatervew auczense), 58 UCCEEDED (aiogram.enums.revenue_withdrawal_state_type.RevenueWitha
sticker_type (aiogram.types.sticker_set.StickerSet at-	attribute), 550
	UCCESSFUL_PAYMENT (aiogram.enums.content_type.ContentType
StickerFormat (class in	attribute), 540
· ·	uccessful_payment (aiogram.types.message.Message
stickers (aiogram.methods.create_new_sticker_set.CreateN	
attribute), 358	uccessfulPayment (class in
stickers (aiogram.types.sticker_set.StickerSet at-	aiogram.types.successful_payment), 334
	uggested_tip_amounts
StickerSet (class in aiogram.types.sticker_set), 343	(aiogram.methods.create_invoice_link.CreateInvoiceLink
StickerType (class in aiogram.enums.sticker_type), 551	attribute), 524
_	uggested_tip_amounts
(aiogram.types.message.Message method), 250	(aiogram.methods.send_invoice.SendInvoice
stop_polling() (aiogram.dispatcher.dispatcher.Dispatches	attribute), 529
method), 566	(aiogram.types.input_invoice_message_content.InputInvoiceMess
StopMessageLiveLocation (class in	attribute), 321
	UPERGROUP (aiogram.enums.chat_type.ChatType
507	attribute), 539
StopPol1 (class in aiogram.methods.stop_poll), 509	UPERGROUP_CHAT_CREATED
STORY (aiogram.enums.content_type.ContentType at-	(aiogram.enums.content_type.ContentType
tribute), 540	attribute), 540
$\verb story (a iogram.types.external_reply_info.ExternalReplyInfo.e$	upergroup_chat_created
attribute), 132	(aiogram.types.message.Message attribute),
story (aiogram.types.message.Message attribute), 202	204
	upports_inline_queries (aiogram.types.user.User
stream_content() (aiogram.client.session.base.BaseSessio	
	upports_streaming(aiogram.methods.send_video.SendVideo
<pre>street_line1(aiogram.types.shipping_address.ShippingAa attribute), 331</pre>	ldress attribute), 467 upports_streaming(aiogram.types.input_media_video.InputMediaVide
street_line2 (aiogram.types.shipping_address.ShippingAa	
	upports_streaming(aiogram.types.input_paid_media_video.InputPaid.
STRIKETHROUGH (aiogram.enums.message_entity_type.Messa	
	witch_inline_query
Strikethrough (class in aiogram.utils.formatting), 657	(aiogram.types.inline_keyboard_button.InlineKeyboardButton
subscription_expiration_date	attribute), 181
(aiogram.types.successful_payment.SuccessfulPays	
attribute), 334	(aiogram.types.inline_keyboard_button.InlineKeyboardButton
subscription_period	attribute), 181
(aiogram.methods.create_chat_subscription_invites	
attribute), 393	(aiogram.types.inline_keyboard_button.InlineKeyboardButton
subscription_period	attribute), 181
(aiogram.methods.create_invoice_link.CreateInvois	
attribute), 523 subscription_period	(aiogram.methods.answer_inline_query.AnswerInlineQuery attribute), 511
	witch_pm_text(aiogram.methods.answer_inline_query.AnswerInlineQu
attribute), 58	attribute), 511
	witchInlineQueryChosenChat (class in
	onPartneallogram.types.switch_inline_query_chosen_chat),
attribute), 339	269

attribute), 339

Т	text (aiogram.types.keyboard_button.KeyboardButton
TELEGRAM_ADS (aiogram.enums.transaction_partner_type.attribute), 552	Transaction Faither Type 1 text (aiogram.types.menu_button.MenuButton at-
TELEGRAM_API (aiogram.enums.transaction_partner_type.attribute), 552	
telegram_payment_charge_id	attribute), 199
(ajogram methods edit user star subscription F	die Sterstans types message. Message attribute), 202
attribute), 525	text (aiogram.types.poll_option.PollOption attribute),
talagram naument charge id	264
(ajogram methods refund star payment Refunds	t toxt (ajogram.types.text_quote.TextQuote attribute), 270
attribute), 527	text (aiogram.utils.callback_answer.CallbackAnswer
talagram naument charge id	property), 651
(aiogram.types.refunded_payment.RefundedPaym	"Text (class in aiogram.utils.formatting), 655
attribute), 329	text_entities (aiogram.methods.send_gift.SendGift
telegram_payment_charge_id	attribute), 367
(aiogram types successful payment Successful Pa	ytext_entities (aiogram.types.game.Game attribute),
attribute), 334	356
TelegramAPIError, 618	text_entities (aiogram.types.input_poll_option.InputPollOption
TelegramAPIServer (class in aiogram.client.telegram),	attribute), 191
14	text_entities (aiogram.types.poll_option.PollOption
TelegramBadRequest, 618	attribute), 264
TelegramConflictError, 618	TEXT_LINK (aiogram.enums.message_entity_type.MessageEntityType
TelegramEntityTooLarge, 618	attribute), 548
	TEXT_MENTION (aiogram.enums.message_entity_type.MessageEntityType
TelegramForbiddenError, 618	attribute), 548
TelegramMigrateToChat, 618 TelegramNetworkError, 618	text_parse_mode(aiogram.methods.send_gift.SendGift
TelegramNotFound, 618	attribute), 367
TelegramRetryAfter, 618	text_parse_mode(aiogram.types.input_poll_option.InputPollOption
TelegramServerError, 618	attribute), 191
TelegramUnauthorizedError, 618	TextLink (class in aiogram.utils.formatting), 658
TEMPORARY_REGISTRATION	TextMention (class in aiogram.utils.formatting), 658
(ajogram anums energynted passport element Fr	cText@passportersinenjogram.types.text_quote), 270
attribute), 545	THB (aiogram.enums.currency.Currency attribute), 544
TEXT (aiogram.enums.content_type.ContentType at-	theme_name (aiogram.types.background_type_chat_theme.BackgroundTyp
tribute), 539	attribute), 22
text (aiogram.filters.command.CommandObject prop-	thumbnail (aiogram.methods.send_animation.SendAnimation
arty) 572	attribute), 433
text(aiogram.methods.answer_callback_query.AnswerCa	hthumbnail (aiogram.methods.send_audio.SendAudio at-
attribute) 380	tribute), 430
tarione), 500	thumbnail (aiogram.methods.send_document.SendDocument
attribute), 506	attribute), 444
text (aiogram.methods.send_gift.SendGift attribute),	thumbnail (aiogram.methods.send_video.SendVideo at-
366	tribute), 467
text (aiogram.methods.send_message.SendMessage at-	thumbnail (aiogram.methods.send_video_note.SendVideoNote
tribute), 452	attribute), 469
text (aiogram.types.copy_text_button.CopyTextButton	thumbnail (aiogram.methods.set_sticker_set_thumbnail.SetStickerSetThumbnail.
attribute), 130	attribute), 375
text (aiogram.types.game.Game attribute), 356	thumbnail (aiogram.types.animation.Animation at-
text (aiogram.types.inline_keyboard_button.InlineKeyboa	
attribute) 181	Chumbhall (diogram.types.dudio.Audio ditribute), 20
text (aiogram.types.inline_query_results_button.InlineQu	ethumbnailutto (aiogram.types.document.Document at-
attribute), 318	tribute), 131
text (aiogram.types.input_poll_option.InputPollOption	thumbnail (aiogram.types.input_media_animation.InputMediaAnimation
attribute), 191	attribute), 184

```
thumbnail (aiogram.types.input_media_audio.InputMediaAhhhirobnail_width (aiogram.types.inline_query_result_location.InlineQue
                      attribute), 185
                                                                                                                                                                  attribute), 307
thumbnail (aiogram.types.input_media_document.InputMethindbnamknth (aiogram.types.inline_query_result_venue.InlineQuery_
                                                                                                                                                                  attribute), 314
                       attribute), 186
thumbnail (aiogram.types.input_media_video.InputMedia\tiilme_zone_name (aiogram.types.business_opening_hours.BusinessOpenin
                      attribute), 188
                                                                                                                                                                  attribute), 30
thumbnail (aiogram.types.input_paid_media_video.InputPaidMediaVideogram.methods.get_updates.GetUpdates_at-
                       attribute), 190
                                                                                                                                                                  tribute), 533
thumbnail (aiogram.types.sticker.Sticker attribute), 342
                                                                                                                                           {\tt title}\ (aiogram.methods.create\_invoice\_link.CreateInvoiceLink
thumbnail
                                  (aiogram.types.sticker_set.StickerSet at-
                                                                                                                                                                  attribute), 523
                      tribute), 343
                                                                                                                                           \verb+title+ (a iogram.methods.create\_new\_sticker\_set.CreateNewStickerSet
thumbnail (aiogram.types.video.Video attribute), 274
                                                                                                                                                                  attribute), 358
thumbnail (aiogram.types.video_note.VideoNote at- title
                                                                                                                                                                 (aiogram.methods.send_audio.SendAudio
                      tribute), 276
                                                                                                                                                                  tribute), 436
thumbnail_height(aiogram.types.inline_query_result_artiitelknli(wiQgwarykkwathtwAtxisulad_invoice.SendInvoice at-
                      attribute), 282
                                                                                                                                                                   tribute), 529
thumbnail_height(aiogram.types.inline_query_result_comiatelf.nline@yeenyResetlt6dontsectd_venue.SendVenue at-
                      attribute), 301
                                                                                                                                                                  tribute), 463
thumbnail_height (aiogram.types.inline_query_result_docinteen.laling@merrellesodktDocahan_title.SetChatTitle at-
                                                                                                                                                                  tribute), 480
                       attribute), 303
thumbnail_height (aiogram.types.inline_query_result_locatedne_hallionerQuernyRbsudsIsot_astioker_set_title.SetStickerSetTitle
                      attribute), 307
                                                                                                                                                                  attribute), 376
thumbnail_height (aiogram.types.inline_query_result_vertice.Irel(vaioQureuryRepublidendium.Audio attribute), 20
                      attribute), 314
                                                                                                                                           title (aiogram.types.business intro.BusinessIntro at-
thumbnail_mime_type
                                                                                                                                                                  tribute), 29
                      (aiogram.types.inline_query_result_gif.InlineQueryResal(Giifgram.types.chat.Chat attribute), 33
                      attribute), 305
                                                                                                                                           title (aiogram.types.chat_full_info.ChatFullInfo at-
thumbnail_mime_type
                                                                                                                                                                  tribute), 55
                      (aiogram.types.inline_query_result_mpeg4_gif.InlineQaeryRasahtMpeg4Gifchat_shared.ChatShared
                                                                                                                                                                                                                                                                          at-
                      attribute), 310
                                                                                                                                                                  tribute), 129
thumbnail_url (aiogram.types.inline_query_result_articlet.htldre:QuiogyRoxthlp\textigdene.Game attribute), 356
                      attribute), 281
                                                                                                                                           thumbnail_url(aiogram.types.inline_query_result_contact.InlineQuatryiRuse))tC8htact
                       attribute), 301
                                                                                                                                           title(aiogram.types.inline_query_result_audio.InlineQueryResultAudio
thumbnail_url(aiogram.types.inline_query_result_document.InlineQtrabvtResultDocument
                      attribute), 303
                                                                                                                                           title(aiogram.types.inline_query_result_cached_document.InlineQueryR
thumbnail_url(aiogram.types.inline_query_result_gif.InlineQueryRatrihGtif), 287
                       attribute), 305
                                                                                                                                           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.InlineQuueri/Rasi).4fl&cation
                                                                                                                                           \verb|title| (a iogram.types.in line\_query\_result\_cached\_mpeg4\_gif.In lineQuery_result\_cached\_mpeg4\_gif.In lineQuery_result\_cached_mpeg4\_gif.In lineQuery_result\_cached_mpeg4\_gif.In lineQuery_result\_cached_mpeg4\_gif.In lineQuery_result\_cached_mpeg4\_gif.In lineQuery_result\_cached_mpeg4\_gif.In lineQuery_result\_cached_mpeg4\_gif.In lineQuery_result\_
                       attribute), 307
thumbnail_url(aiogram.types.inline_query_result_mpeg4_gif.Inline@pribnytResultMpeg4Gif
                       attribute), 310
                                                                                                                                           title(aiogram.types.inline_query_result_cached_photo.InlineQueryResult_
thumbnail_url(aiogram.types.inline_query_result_photo.InlineQueantRibsudleP,hote
                      attribute), 312
                                                                                                                                           \verb|title| (a iogram.types.in line\_query\_result\_cached\_video.In lineQueryResult\_cached\_video.In lineQueryResult\_cached\_video.I
thumbnail_url(aiogram.types.inline_query_result_venue.InlineQueartiRibsult),e2016e
                      attribute), 314
                                                                                                                                           title(aiogram.types.inline_query_result_cached_voice.InlineQueryResult
thumbnail_url(aiogram.types.inline_query_result_video.InlineQuewyRedsuleVideo
                      attribute), 316
                                                                                                                                           title(aiogram.types.inline_query_result_document.InlineQueryResultDoc
thumbnail_width(aiogram.types.inline_query_result_article.InlineQtteibyRe)ullArticle
                      attribute), 282
                                                                                                                                           thumbnail_width(aiogram.types.inline_query_result_contact.Inline@trebyRe)sidloContact
                      attribute), 301
                                                                                                                                           {\tt title}\ (aiogram.types.inline\_query\_result\_location.InlineQueryResultLocation)
thumbnail_width(aiogram.types.inline_query_result_document.Inlinut@burteyResMtDocument
                      attribute), 303
                                                                                                                                           title(aiogram.types.inline_query_result_mpeg4_gif.InlineQueryResultMy
```

attribute), 310	aiogram.types.transaction_partner_fragment),
$\verb title (a iogram.types.inline_query_result_photo.InlineQuery_result$	
attribute), 312	TransactionPartnerOther (class in
title (aiogram.types.inline_query_result_venue.InlineQuattribute), 313	eryResultV aing ram.types.transaction_partner_other), 337
$\verb title (a iogram.types.inline_query_result_video.InlineQuery_result$	erTy RansktVtd.co nPartnerTelegramAds (class in
attribute), 316	<pre>aiogram.types.transaction_partner_telegram_ads),</pre>
title(aiogram.types.inline_query_result_voice.InlineQue	
attribute), 317	TransactionPartnerTelegramApi (class in
attribute), 320	atInvoiceM esisaggeGotypus .transaction_partner_telegram_api), 338
$\verb title (a iogram.types.input_media_audio.InputMediaAudio) $	- · · · · · · · · · · · · · · · · · · ·
attribute), 185	aiogram.enums.transaction_partner_type),
title(aiogram.types.input_venue_message_content.Input	
attribute), 324	TransactionPartnerUser (class in
title (aiogram.types.invoice.Invoice attribute), 326	aiogram.types.transaction_partner_user),
title (aiogram.types.shipping_option.ShippingOption	339
attribute), 331 title (aiogram.types.sticker_set.StickerSet attribute),	transactions (aiogram.types.star_transactions.StarTransactions attribute), 334
343	${\tt translation} \ (a iogram. types. encrypted_passport_element. Encrypted Passet and the passport_element. The passport_element and the passport_element and the passport_element. The passport_element and the passport_element and the passport_element and the passport_element. The passport_element and the passport_eleme$
title (aiogram.types.venue.Venue attribute), 273	attribute), 345
title (aiogram.utils.web_app.WebAppChat attribute), 648	TRANSLATION_FILE (aiogram.enums.passport_element_error_type.Passpattribute), 550
TJS (aiogram.enums.currency.Currency attribute), 544	TRANSLATION_FILES (aiogram.enums.passport_element_error_type.Pass
TokenBasedRequestHandler (class in	attribute), 550
aiogram.webhook.aiohttp_server), 623	${\tt traveler} (a iogram. types. proximity_alert_triggered. ProximityAlertTriggered. ProximityAlertAlertAlertAlertAlertAlertAlertAlert$
top_color(aiogram.types.background_fill_gradient.Background_fill_gradie	
attribute), 21	TRY (aiogram.enums.currency.Currency attribute), 544
	TTD (aiogram.enums.currency.Currency attribute), 544
aiogram.enums.topic_icon_color), 551	TWD (aiogram.enums.currency.Currency attribute), 544
total_amount (aiogram.types.invoice.Invoice attribute), 326	type (aiogram.methods.send_poll.SendPoll attribute), 461
total_amount (aiogram.types.pre_checkout_query.PreCh attribute), 328	e tkant Quegr am.types.background_fill_freeform_gradient.BackgroundFillFi attribute), 21
total_amount (aiogram.types.refunded_payment.Refunde attribute), 328	d Pytym (aniogram.types.background_fill_gradient.BackgroundFillGradient attribute), 21
· · · · · · · · · · · · · · · · · · ·	sfulPa miergram.types.background_fill_solid.BackgroundFillSolid
	attribute), 22
total_count (aiogram.types.gift.Gift attribute), 340	type (aiogram.types.background_type_chat_theme.BackgroundTypeChatT
total_count(aiogram.types.reaction_count.ReactionCou	
attribute), 264	type (aiogram.types.background_type_fill.BackgroundTypeFill
total_count(aiogram.types.user_profile_photos.UserPro	ofilePhotos attribute), 23
attribute), 272	type (aiogram.types.background_type_pattern.BackgroundTypePattern
<pre>total_voter_count (aiogram.types.poll.Poll attribute),</pre>	attribute), 23
262	${\tt type} \ (a iogram. types. background_type_wall paper. Background Type Wall Paper.$
TransactionPartner (class in	attribute), 24
aiogram.types.transaction_partner), 335	${\tt type} \ (aiogram.types.bot_command_scope_all_chat_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_scope_administrators.BotCommand_$
TransactionPartnerAffiliateProgram (class in	attribute), 25
aiogram.types.transaction_partner_affiliate_prog 335	gr typp); (aiogram.types.bot_command_scope_all_group_chats.BotCommand attribute), 26
TransactionPartnerChat (class in	type(aiogram.types.bot_command_scope_all_private_chats.BotComman
aiogram.types.transaction_partner_chat),	attribute), 26
336	$\verb"type" (a iogram.types.bot_command_scope_chat.BotCommandScopeChat") \\$
TransactionPartnerFragment (class in	attribute), 26

- type (aiogram.types.bot_command_scope_chat_administrators.BotCattmibutts)cope_ChatAdministrators

 attribute), 27

 type (aiogram.types.input_media_audio.InputMediaAudio

 type (aiogram.types.bot_command_scope_chat_member.BotCommandsripac)attribute), 27

 type (aiogram.types.input_media_document.InputMediaDocument
- type (aiogram.types.bot_command_scope_default.BotCommandScopetDefbute), 186
 attribute), 28
 type (aiogram.types.input_media_photo.InputMediaPhoto
- type (aiogram.types.chat.Chat attribute), 33

 attribute), 187
- type (aiogram.types.chat_background.ChatBackground type (aiogram.types.input_media_video.InputMediaVideo attribute), 50 attribute), 188
- type (aiogram.types.chat_full_info.ChatFullInfo at-type(aiogram.types.input_paid_media_photo.InputPaidMediaPhoto tribute), 54 attribute), 189
- type (aiogram.types.encrypted_passport_element.Encrypted.passpoidElametypes.input_paid_media_video.InputPaidMediaVideo attribute), 344 attribute), 190
- type (aiogram.types.inline_query_result_article.InlineQueryResultArgicle.types.keyboard_button_poll_type.KeyboardButtonPollType attribute), 281 attribute), 192
- type (aiogram.types.inline_query_result_audio.InlineQueryRepaltAudiiogram.types.menu_button.MenuButton atattribute), 282 tribute), 198
- type (aiogram.types.inline_query_result_cached_audio.Inli**tryQuent/Quent/Result/Cypetresult**@utton_commands.MenuButtonCommands attribute), 284 attribute), 198
- type (aiogram.types.inline_query_result_cached_documentstypine@ueryResyptGanhad_Documentedefault.MenuButtonDefault attribute), 287 attribute), 199
- type (aiogram.types.inline_query_result_cached_gif.Inline@ypeyRisught@ndhypdGiffenu_button_web_app.MenuButtonWebApp attribute), 288 attribute), 199
- type (aiogram.types.inline_query_result_cached_mpeg4_gif_yndene@iagraRustypexachesslylpeg#Gif.MessageEntity atattribute), 291 tribute), 255
- type (aiogram.types.inline_query_result_cached_photo.Inli**nyQueeniResult.Cypdsedlehsage**_origin_channel.MessageOriginChannel attribute), 294

 attribute), 294
- type (aiogram.types.inline_query_result_cached_sticker.Inl**trytQuentyResultGpek.ad&siclger_**origin_chat.MessageOriginChat attribute), 296 attribute), 257
- type (aiogram.types.inline_query_result_cached_video.Inli**ntyQueryiRgsauhtGypehe.d\idee**corigin_hidden_user.MessageOriginHiddenU
 attribute), 298
 attribute), 258

 type (aiogram types inline_query_result_cached_video.InlintyQueryiRgsauhtGypehed\ideecorigin_hidden_user.MessageOriginHiddenU
- type (aiogram.types.inline_query_result_cached_voice.Inlinte@pedxyRgsultGyphedtVexiswege_origin_user.MessageOriginUser attribute), 299 attribute), 258
- type (aiogram.types.inline_query_result_contact.InlineQue**tyRes(diGgnunct**ypes.paid_media_photo.PaidMediaPhoto attribute), 301 attribute), 261
- type (aiogram.types.inline_query_result_document.InlineQtypeRaxingPontypestpaid_media_preview.PaidMediaPreview attribute), 302 attribute), 261
- type (aiogram.types.inline_query_result_game.InlineQuery**Repul**(**@ioge**am.types.paid_media_video.PaidMediaVideo attribute), 303 attribute), 261
- type (aiogram.types.inline_query_result_gif.InlineQueryRetypeGfaiogram.types.passport_element_error_data_field.PassportElementEntribute), 304

 attribute), 346
- type (aiogram.types.inline_query_result_location.InlineQu**tryyPexaltlypantityp**es.passport_element_error_file.PassportElementErrorFile attribute), 306 attribute), 347
- type (aiogram.types.inline_query_result_mpeg4_gif.Inline@vypyReiogltMpayfesifassport_element_error_files.PassportElementErrorF attribute), 310 attribute), 348
- type (aiogram.types.inline_query_result_photo.InlineQuery**Repal(dribgma**m.types.passport_element_error_front_side.PassportElementE attribute), 311 attribute), 348
- type (aiogram.types.inline_query_result_venue.InlineQuerytherettlement_attribute), 313 attribute), 349
- type (aiogram.types.inline_query_result_video.InlineQueryRepal(Nidgram.types.passport_element_error_selfie.PassportElementErrorSattribute), 315

 attribute), 350
- type (aiogram.types.inline_query_result_voice.InlineQuery**Rypel(Vioige**am.types.passport_element_error_translation_file.PassportElement_attribute), 317

 attribute), 350
- type (aiogram.types.input media animation.InputMediaAntmptioniogram.types.passport element error translation files.PassportEle

attribute), 351	unclaimed_prize_count
	PassportEl &micryfEurnotylpaspgivifund ay_completed.GiveawayCompleted attribute), 137
type (aiogram.types.poll.Poll attribute), 262	unclaimed_prize_count
type (aiogram.types.reaction_count.ReactionCount attribute), 264	(aiogram.types.giveaway_winners.GiveawayWinners attribute), 138
<pre>type (aiogram.types.reaction_type_custom_emoji.Reaction</pre>	(I)NDERISINELmojjram.enums.message_entity_type.MessageEntityType attribute), 548
$\verb"type" (a iogram.types.reaction_type_emoji.ReactionTypeEmoji.Re$	
attribute), 265	UnhideGeneralForumTopic (class in
type (aiogram.types.reaction_type_paid.ReactionTypePaid attribute), 266	aiogram.methods.unhide_general_forum_topic), 491
type (aiogram.types.revenue_withdrawal_state_failed.Reveatribute), 329	e WNXWWN rdwial &tataEmiled s.content_type.ContentType at- tribute), 539
type (aiogram.types.revenue_withdrawal_state_pending.R attribute), 330	e værpæðki(i) dr ímiokStane.fletedi mgallback_data.CallbackData class method), 578
$\verb"type" (a iogram.types.revenue_with drawal_state_succeeded") \\$.Respins(W(thidgorumlSyptesSmessagedMessage method), 252
attribute), 330	<pre>unpin_all_general_forum_topic_messages()</pre>
type (aiogram.types.sticker.Sticker attribute), 342	(aiogram.types.chat.Chat method), 48
type (aiogram.types.transaction_partner_affiliate_program	
attribute), 336	method), 42
type (aiogram.types.transaction_partner_chat.Transaction_attribute), 336	42
<pre>type (aiogram.types.transaction_partner_fragment.Transa</pre>	$aiogram.methods.unpin_all_chat_messages),$
type (aiogram.types.transaction_partner_other.Transaction_ attribute), 337	UnpinAllForumTopicMessages (class in
type (aiogram.types.transaction_partner_telegram_ads.Transaction_p	ansactionP aitigeriFalisynthiodAsdis npin_all_forum_topic_messages), 493
$\verb"type" (a iogram.types.transaction_partner_telegram_api.Transaction_partner_telegram_api.Transaction_partner_telegram_api.Transaction_partner_telegram_api.Transaction_partner_telegram_api.Transaction_partner_telegram_api.Transaction_partner_telegram_api.Transaction_a$	uWnpinduPhGenefal/GranumAPopicMessages (class in
attribute), 338	$aiogram.methods.unpin_all_general_forum_topic_messages),$
${\tt type} \ (aiogram.types.transaction_partner_user.Transaction_user.Transa$	
attribute), 339	UnpinChatMessage (class in
type (aiogram.utils.web_app.WebAppChat attribute),	aiogram.methods.unpin_chat_message), 495
648 TYPING (aiogram.enums.chat action.ChatAction at-	unrestrict_boost_count (aiogram.types.chat.Chat
TYPING (aiogram.enums.chat_action.ChatAction attribute), 538	<pre>attribute), 37 unrestrict_boost_count</pre>
typing() (aiogram.utils.chat_action.ChatActionSender	(aiogram.types.chat_full_info.ChatFullInfo
class method), 644	attribute), 56
TZS (aiogram.enums.currency.Currency attribute), 544	UNSPECIFIED (aiogram.enums.passport_element_error_type.PassportElemattribute), 550
U	UnsupportedKeywordArgument, 618
UAH (aiogram.enums.currency.Currency attribute), 544	until_date(aiogram.methods.ban_chat_member.BanChatMember
UGX (aiogram.enums.currency.Currency attribute), 544	attribute), 382
unban() (aiogram.types.chat.Chat method), 46	until_date(aiogram.methods.restrict_chat_member.RestrictChatMember
unban_sender_chat() (aiogram.types.chat.Chat	attribute), 429
method), 38	$\verb"until_date" (a iogram. types. chat_member_banned. Chat Member Banned$
UnbanChatMember (class in	attribute), 103
aiogram.methods.unban_chat_member), 489	until_date(aiogram.types.chat_member_member.ChatMemberMember attribute), 103
UnbanChatSenderChat (class in	$\verb"until_date" (a iogram. types. chat_member_restricted. Chat Member Restricted and the property of the prope$
aiogram.methods.unban_chat_sender_chat),	attribute), 105
490	update (aiogram.types.error_event.ErrorEvent at- tribute), 617

Update (class in aiogram.types.update), 353	277
update_data() (aiogram.fsm.scene.SceneWizard method), 613	url (aiogram.types.webhook_info.WebhookInfo at- tribute), 355
<pre>update_data() (aiogram.fsm.storage.base.BaseStorage</pre>	url (aiogram.utils.callback_answer.CallbackAnswer property), 651
<pre>update_handler_flags() (aiogram.filters.base.Filter</pre>	Url (class in aiogram.utils.formatting), 656
method), 582	URLInputFile (class in aiogram.types.input_file), 183,
update_id(aiogram.types.update.Update attribute), 353	556
UpdateType (class in aiogram.enums.update_type), 552	USD (aiogram.enums.currency.Currency attribute), 544
UpdateTypeLookupError, 355	use_independent_chat_permissions
upgrade_star_count (aiogram.types.gift.Gift attribute), 340	(aiogram.methods.restrict_chat_member.RestrictChatMember attribute), 429
UPLOAD_DOCUMENT (aiogram.enums.chat_action.ChatActio	
attribute), 538	$(a iogram.methods.set_chat_permissions.SetChatPermissions$
${\tt upload_document()} \ (a iogram.utils.chat_action.ChatAction) \ (a iogram.utils.chat_action) \ (a iogram.utils.chat$	
class method), 644	USER (aiogram.enums.message_origin_type.MessageOriginType
UPLOAD_PHOTO (aiogram.enums.chat_action.ChatAction	attribute), 548
attribute), 538 upload_photo() (aiogram.utils.chat_action.ChatActionSe	USER (aiogram.enums.transaction_partner_type.TransactionPartnerType ender attribute), 552
class method), 644	${\tt user} \ (aiogram. types. business_connection. Business Connection$
${\tt UPLOAD_VIDEO}\ (a iogram.enums.chat_action.ChatAction$	attribute), 29
attribute), 538	$user ({\it aiogram.types.chat_boost_source_gift_code.ChatBoostSourceGiftCollege}) \\$
${\tt upload_video()} \ (a iogram.utils.chat_action.ChatActionSet \ and \$	
class method), 645	$user ({\it aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGiveaway}) and {\it aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGiveaway}) and {\it aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGiveaway}) and {\it aiogram.types.chat_boost_source_giveaway.ChatBoostSourceGiveaway}) and {\it aiogram.types.chat_boost_source_giveaway}. A constant {\it aiogram.types.chat_boost_sour$
${\tt UPLOAD_VIDEO_NOTE}\ (aiogram.enums.chat_action.Chat Action.Chat Action.Ch$	
attribute), 538	$user ({\it aiogram.types.chat_boost_source_premium.ChatBoostSourcePr$
<pre>upload_video_note()</pre>	attribute), 52
(aiogram.utils.chat_action.ChatActionSender class method), 645	user (aiogram.types.chat_member_administrator.ChatMemberAdministrator.thatMemb
UPLOAD_VOICE (aiogram.enums.chat_action.ChatAction attribute), 538	user (aiogram.types.chat_member_banned.ChatMemberBanned attribute), 103
$\verb"upload_voice"() (a iogram.utils.chat_action.ChatActionSet and action.ChatActionSet action.ChatAct$	e nder (aiogram.types.chat_member_left.ChatMemberLeft
class method), 645	attribute), 103
UploadStickerFile (class in	${\tt user}(aiogram.types.chat_member_member.ChatMemberMember$
<pre>aiogram.methods.upload_sticker_file), 377</pre>	attribute), 103
<pre>URL (aiogram.enums.message_entity_type.MessageEntityTy attribute), 548</pre>	puser (aiogram.types.chat_member_owner.ChatMemberOwner attribute), 104
url (aiogram.methods.answer_callback_query.AnswerCallatribute), 380	busk@(uciogram.types.chat_member_restricted.ChatMemberRestricted attribute), 104
url (aiogram.methods.set_webhook.SetWebhook attribute), 534	user (aiogram.types.game_high_score.GameHighScore attribute), 357
url (aiogram.types.inline_keyboard_button.InlineKeyboard attribute), 181	dBatton (aiogram.types.message_entity.MessageEntity at- tribute), 256
url (aiogram.types.inline_query_result_article.InlineQuerattribute), 281	y Real (Airigham. types. message_reaction_updated. Message Reaction Updated attribute), 260
url (aiogram.types.link_preview_options.LinkPreviewOpti attribute), 196	
url (aiogram.types.login_url.LoginUrl attribute), 197	user(aiogram.types.transaction_partner_user.TransactionPartnerUser
url (aiogram.types.message_entity.MessageEntity attribute), 256	attribute), 339 user (aiogram.utils.web_app.WebAppInitData attribute),
url (aiogram.types.revenue_withdrawal_state_succeeded.I	
attribute), 331	User (class in aiogram.types.user), 270
url (aiogram.types.user.User property), 271	user_administrator_rights
url (aiogram.types.web_app_info.WebAppInfo attribute),	(aiogram.types.keyboard_button_request_chat.KeyboardButtonRe

attribute), 194		attribute), 379		
user_chat_id(aiogram.types.business_connection.Busin attribute), 29		tiaio gram.types.bot_commar attribute), 27	ıd_scope_chat_n	nember.BotCommand
<pre>user_chat_id(aiogram.types.chat_join_request.ChatJoin</pre>	n <i>Rossporesi</i> id	(aiogram.types.contact.Con	tact attribute), 1'	29
attribute), 58	_	(aiogram.types.shared_us		
user_id(aiogram.methods.add_sticker_to_set.AddSticker		tribute), 268		
attribute), 357		(aiogram.types.user_share	ed.UserShared	at-
user_id(aiogram.methods.approve_chat_join_request.Ap				
attribute), 381		s (aiogram.types.users_sha	red.UsersShared	at-
user_id(aiogram.methods.ban_chat_member.BanChatMe		tribute), 273		
attribute), 382		_CHAT (aiogram.fsm.strateg	gy.FSMStrategy	at-
user_id(aiogram.methods.create_new_sticker_set.Create				
attribute), 358		_TOPIC (aiogram.fsm.strate	gy.FSMStrategy	at-
user_id(aiogram.methods.decline_chat_join_request.Dec				
attribute), 395		_bot (aiogram.types.keyboa	rd button reque	st user.KeyboardBut
user_id(aiogram.methods.edit_user_star_subscription.Ea			1	_ ,
attribute), 525		_bot (aiogram.types.keyboa	rd button reque	st users.KevboardBu
user_id(aiogram.methods.get_chat_member.GetChatMer		attribute), 195	1	_ ,
attribute), 411		_premium(aiogram.types.ke	yboard button	request user.Keyboar
user_id(aiogram.methods.get_game_high_scores.GetGan			<i>y</i>	1=
attribute), 516		_premium(aiogram.types.ke	evboard button	reauest users.Kevboo
user_id(aiogram.methods.get_user_chat_boosts.GetUser			,	-4
attribute), 419		ARED (aiogram.enums.conter	nt_type.ContentT	vne
user_id(aiogram.methods.get_user_profile_photos.GetU.				Jr -
attribute), 420	user_sha		sage.Message	at-
user_id(aiogram.methods.promote_chat_member.Promo				
attribute), 425	UserCha		iss	in
user_id(aiogram.methods.refund_star_payment.RefundS	tarPaymeni	taiogram.types.user chat be	oosts), 272	
attribute), 527		e (aiogram.types.chat.Chat a		
user_id(aiogram.methods.remove_user_verification.Rem				at-
attribute), 364		tribute), 55	J	
user_id(aiogram.methods.replace_sticker_in_set.Replace			ed.ChatShared	at-
attribute), 365		tribute), 129		
user_id(aiogram.methods.restrict_chat_member.Restrict	ChseMnada	er (aiogram.types.shar	ed_user.SharedU	ser
attribute), 429		attribute), 268		
user_id(aiogram.methods.save_prepared_inline_message	e.Seeckhaan	ei k edbigbiawiMyysea.ge ser. User d	attribute), 270	
attribute), 515	username			at-
user_id(aiogram.methods.send_gift.SendGift attribute),		tribute), 648	• •	
366	username	e (aiogram.utils.web_app	.WebAppUser	at-
user_id(aiogram.methods.set_chat_administrator_custor.	n_title.SetC			
attribute), 474	UserPro	filePhotos (a	class	in
user_id(aiogram.methods.set_game_score.SetGameScore	e	aiogram.types.user_profile_	<i>photos</i>), 272	
attribute), 519	users	(aiogram.types.users_shared	d.UsersShared	at-
user_id(aiogram.methods.set_passport_data_errors.SetF	PassportDa	tta:Hourte)s, 273		
attribute), 536	users (ai	ogram.types.video_chat_pa	rticipants_invited	d.VideoChatParticipa
user_id(aiogram.methods.set_sticker_set_thumbnail.SetS			•	•
attribute), 375		HARED (aiogram.enums.cont	ent_type.Content	Туре
user_id(aiogram.methods.set_user_emoji_status.SetUser				-
attribute), 488	users_sl	nared (aiogram.types.mes	ssage.Message	at-
user_id(aiogram.methods.unban_chat_member.UnbanCh	hatMember	tribute), 204		
attribute), 489		red (class in aiogram.types.	user_shared), 27	12
$\verb"user_id" (a iogram.methods.upload_sticker_file. UploadSticker_file) and the sticker_file of the sticke$	ck⊌sÆideSha	ared (class in aiogram.types	.users_shared),?	273
attribute), 377	UTILITY	_BILL (aiogram.enums.encr	ypted_passport_	element.EncryptedPa
user_id (aiogram.methods.verify_user.VerifyUser		attribute), 545		

UYU (aiogram.enums.currency.Currency attribute), 544 UZS (aiogram.enums.currency.Currency attribute), 544	attribute), 541 video_chat_participants_invited
V	(aiogram.types.message.Message attribute), 205
value (aiogram.types.dice.Dice attribute), 130	VIDEO_CHAT_SCHEDULED
vcard (aiogram.methods.send_contact.SendContact at- tribute), 440	(aiogram.enums.content_type.ContentType attribute), 541
vcard (aiogram.types.contact.Contact attribute), 130	video_chat_scheduled
vcard(aiogram.types.inline_query_result_contact.InlineQuattribute), 301	205
vcard(aiogram.types.input_contact_message_content.Input attribute), 319	ultDhaciMessasteBhfen(aiogram.enums.content_type.ContentType attribute), 541
VENUE (aiogram.enums.content_type.ContentType at- tribute), 540	video_chat_started (aiogram.types.message.Message attribute), 205
attribute), 546	nyideouthyration (aiogram.types.inline_query_result_video.InlineQueryReatribute), 316
venue (aiogram.types.external_reply_info.ExternalReplyInj attribute), 133	gyideo_file_id(aiogram.types.inline_query_result_cached_video.InlineQ attribute), 298
venue (aiogram.types.message.Message attribute), 203 Venue (class in aiogram.types.venue), 273	video_height (aiogram.types.inline_query_result_video.InlineQueryResult_video.
VerifyChat (class in aiogram.methods.verify_chat), 378	VIDEO_NOTE (aiogram.enums.content_type.ContentType
VerifyUser (class in aiogram.methods.verify_user), 379	attribute), 540
via_bot (aiogram.types.message.Message attribute), 202	video_note (aiogram.methods.send_video_note.SendVideoNote attribute), 469
<pre>via_chat_folder_invite_link</pre>	widpganeqte (aiogram.types.external_reply_info.ExternalReplyInfo attribute), 132
via_join_request(aiogram.types.chat_member_updated attribute), 106	
VIDEO (aiogram.enums.content_type.ContentType at- tribute), 540	<pre>video_start_timestamp (aiogram.methods.copy_message.CopyMessage)</pre>
VIDEO (aiogram.enums.inline_query_result_type.InlineQue attribute), 546	video_start_timestamp
VIDEO (aiogram.enums.input_media_type.InputMediaType attribute), 546	(aiogram.methods.forward_message.ForwardMessage attribute), 406
VIDEO (aiogram.enums.input_paid_media_type.InputPaidMattribute), 546	yadag _p url (aiogram.types.inline_query_result_video.InlineQueryResultVi attribute), 316
VIDEO (aiogram.enums.paid_media_type.PaidMediaType attribute), 549	video_width (aiogram.types.inline_query_result_video.InlineQueryResult_attribute), 316
VIDEO (aiogram.enums.sticker_format.StickerFormat at- tribute), 551	VideoChatEnded (class in aiogram.types.video_chat_ended), 275
video (aiogram.methods.send_video.SendVideo at- tribute), 466	VideoChatParticipantsInvited (class in aiogram.types.video_chat_participants_invited),
video (aiogram.types.external_reply_info.ExternalReplyInj attribute), 132	fo 275 VideoChatScheduled (class in
video (aiogram.types.message.Message attribute), 203	aiogram.types.video_chat_scheduled), 275
video (aiogram.types.paid_media_video.PaidMediaVideo attribute), 261	VideoChatStarted (class in aiogram.types.video_chat_started), 275
Video (class in aiogram.types.video), 274	VideoNote (class in aiogram.types.video_note), 276
VIDEO_CHAT_ENDED (aiogram.enums.content_type.Content attribute), 541	HJQLET (aiogram.enums.topic_icon_color.TopicIconColor attribute), 551
video_chat_ended (aiogram.types.message.Message	VND (aiogram.enums.currency.Currency attribute), 544
attribute), 205 VIDEO_CHAT_PARTICIPANTS_INVITED	VOICE (aiogram.enums.content_type.ContentType attribute), 540
(aiogram.enums.content_type.ContentType	${\tt VOICE}(aiogram.enums.inline_query_result_type.InlineQueryResultType$

voice	attribute), 546 (aiogram.methods.send_voice.SendVoice at- tribute), 472	width (aiogram.types.animation.Animation attribute), 19 width (aiogram.types.input_media_animation.InputMediaAnimation attribute), 184
voice(a		nfwidth(aiogram.types.input_media_video.InputMediaVideo attribute), 189
	iogram.types.message.Message attribute), 203 lass in aiogram.types.voice), 276	width(aiogram.types.input_paid_media_video.InputPaidMediaVideo attribute), 190
voice_d	uration(aiogram.types.inline_query_result_voi attribute), 317	ic eงโหน้หนัง QuiogyหิงยระปุPeัoipa id_media_preview.PaidMediaPreview attribute), 261
	attribute), 300	ne sh<u>i</u>vdrib e. Kding Qune.ty Resufth6t ac <u>l</u> wid&& lo toSize attribute), 262
voice_u		ne Query Rasago Vanict ypes.sticker.Sticker attribute), 342
	attribute), 317	width (aiogram.types.video.Video attribute), 274
	tribute), 263	winner_count (aiogram.types.giveaway.Giveaway at- tribute), 136
voter_c		winner_count (aiogram.types.giveaway_completed.GiveawayCompleted
	tribute), 264	attribute), 137
W		winner_count (aiogram.types.giveaway_winners.GiveawayWinners attribute), 138
	attribute), 138	а ууун) ет ş (aiogram.types.giveaway_winners.GiveawayWinners attribute), 138
	$(aiogram.types.proximity_alert_triggered.Proximattribute), 264$	(aiogram.types.giveaway.Giveaway attribute),
	(aiogram.enums.menu_button_type.MenuButton's attribute), 547	winners_selection_date
	attribute), 181	yboardButtoqiogram.types.giveaway_winners.GiveawayWinners attribute), 138
	attribute), 318	ne Dit her Residus But as te (aiogram.types.transaction_partner_fragment.Transac attribute), 337
web_app	(aiogram.types.keyboard_button.KeyboardButton attribute), 192	n wrap_local_file(aiogram.client.telegram.TelegramAPIServer attribute), 15
web_app	(aiogram.types.menu_button.MenuButton attribute), 198	WRITE_ACCESS_ALLOWED (aiogram.enums.content_type.ContentType
web_app	(aiogram.types.menu_button_web_app.MenuBut attribute), 199	tonWebAppattribute), 540 write_access_allowed
WEB_APP	_DATA (aiogram.enums.content_type.ContentType attribute), 541	e (aiogram.types.message.Message attribute), 204
web_app	_data (aiogram.types.message.Message at-tribute), 206	WriteAccessAllowed (class in aiogram.types.write_access_allowed), 277
web_app	_name (aiogram.types.write_access_allowed.Write_attribute), 277	teAccessAllowed
web_app		_queshAfisweiWerloupp@ae.myask_position.MaskPosition at- tribute), 341
WebAppCl	hat (class in aiogram.utils.web_app), 648	V
WebAppD	ata (class in aiogram.types.web_app_data), 277	Υ
	nfo (class in aiogram.types.web_app_info), 277 nitData (class in aiogram.utils.web_app), 647	<pre>y_shift (aiogram.types.mask_position.MaskPosition at- tribute), 341</pre>
WebAppU	ser (class in aiogram.utils.web_app), 648	year (aiogram.types.birthdate.Birthdate attribute), 24
Webhook	<pre>Info (class in aiogram.types.webhook_info), 355</pre>	YELLOW (aiogram.enums.topic_icon_color.TopicIconColor attribute), 551
width(a	iogram.methods.send_animation.SendAnimation attribute), 433	YER (aiogram.enums.currency.Currency attribute), 544
width	(aiogram.methods.send_video.SendVideo at-	Z
	tribute), 466	ZAR (aiogram.enums.currency.Currency attribute), 544