Telegram Bot API

> The Bot API is an HTTP-based interface created for developers keen on building bots for Telegram.
> To learn how to create and set up a bot, please consult our Introduction to Bots and Bot FAQ.

Recent changes

August 27, 2018

Bot API 4.1

- Added support for translated versions of documents in Telegram Passport. New field *translation* in EncryptedPassportElement.
- New errors: PassportElementErrorTranslationFile and PassportElementErrorTranslationFiles, PassportElementErrorUnspecified.

July 26, 2018

Bot API 4.0.

- Added support for **Telegram Passport**. See the official announcement on the blog and the manual for details.
- Added support for **editing the media content of messages**: added the method editMessageMedia and new types InputMediaAnimation, InputMediaAudio, and InputMediaDocument.
- Added the field *thumb* to the Audio object to contain the thumbnail of the album cover to which the music file belongs.
- Added support for attaching custom thumbnails to uploaded files. For animations, audios, videos and video notes, which are less than 10 MB in size, thumbnails are generated automatically.
- `tg://` URLs now can be used in inline keyboard url buttons and `text_link` message entities.
- Added the method sendAnimation, which can be used instead of sendDocument to send animations, specifying their duration, width and height.
- Added the field animation to the Message object. For backward compatibility, when this field is set, the *document* field will be also set.
- Added two new MessageEntity types: *cashtag* and *phone_number*.
- Added support for Foursquare venues: added the new field *foursquare_type* to the objects Venue, InlineQueryResultVenue and InputVenueMessageContent, and the parameter *foursquare_type* to the sendVenue method.
- You can now create inline mentions of users, who have pressed your bot's callback buttons.
- You can now use the `Retry-After` response header to configure the delay after which the Bot API will retry the request after an unsuccessful response from a webhook.
- If a webhook returns the HTTP error `410 Gone` for all requests for more than 23 hours successively, it can be automatically removed.
- Added vCard support when sharing contacts: added the field *vcard* to the objects Contact, InlineQueryResultContact, InputContactMessageContent and the method sendContact.

See earlier changes »

Authorizing your bot

Each bot is given a unique authentication token when it is created. The token looks something like `123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11`, but we'll use simply <token> in this document instead. You can learn about obtaining tokens and generating new ones in this document.

Making requests

All queries to the Telegram Bot API must be served over HTTPS and need to be presented in this form: `https://api.telegram.org/bot<token>/METHOD_NAME`. Like this for example:

```
https://api.telegram.org/bot123456:ABC-DEF1234ghIkl-zyx57W2v1u123ew11/getMe
```

We support **GET** and **POST** HTTP methods. We support four ways of passing parameters in Bot API requests:

- URL query string
- application/x-www-form-urlencoded
- application/json (except for uploading files)
- multipart/form-data (use to upload files)

The response contains a JSON object, which always has a Boolean field 'ok' and may have an optional String field 'description' with a human-readable description of the result. If 'ok' equals true, the request was successful and the result of the query can be found in the 'result' field. In case of an unsuccessful request, 'ok' equals false and the error is explained in the 'description'. An Integer 'error_code' field is also returned, but its contents are subject to change in the future. Some errors may also have an optional field 'parameters' of the type ResponseParameters, which can help to automatically handle the error.

- All methods in the Bot API are case-insensitive.
- All queries must be made using UTF-8.

Making requests when getting updates

If you're using **webhooks**, you can perform a request to the Bot API while sending an answer to the webhook. Use either *application/json* or *application/x-www-form-urlencoded* or *multipart/form-data* response content type for passing parameters. Specify the method to be invoked in the *method* parameter of the request. It's not possible to know that such a request was successful or get its result.

> Please see our FAQ for examples.

Getting updates

There are two mutually exclusive ways of receiving updates for your bot — the getUpdates method on one hand and Webhooks on the other. Incoming updates are stored on the server until the bot receives them either way, but they will not be kept longer than 24 hours.

Regardless of which option you choose, you will receive JSON-serialized Update objects as a result.

Update

This object represents an incoming update.
At most **one** of the optional parameters can be present in any given update.

Field	Type	Description
update_id	Integer	The update's unique identifier. Update identifiers start from a certain positive number and increase sequentially. This ID becomes especially handy if you're using Webhooks, since it allows you to ignore repeated updates or to restore the correct update sequence, should they get out of order. If there are no new updates for at least a week, then identifier of the next update will be chosen randomly instead of sequentially.
message	Message	*Optional*. New incoming message of any kind — text, photo, sticker, etc.
edited_message	Message	*Optional*. New version of a message that is known to the bot and was edited
channel_post	Message	*Optional*. New incoming channel post of any kind — text, photo, sticker, etc.
edited_channel_post	Message	*Optional*. New version of a channel post that is known to the bot and was edited
inline_query	InlineQuery	*Optional*. New incoming inline query
chosen_inline_result	ChosenInlineResult	*Optional*. The result of an inline query that was chosen by a user and sent to their chat partner. Please see our documentation on the feedback

		collecting for details on how to enable these updates for your bot.
callback_query	CallbackQuery	*Optional*. New incoming callback query
shipping_query	ShippingQuery	*Optional*. New incoming shipping query. Only for invoices with flexible price
pre_checkout_query	PreCheckoutQuery	*Optional*. New incoming pre-checkout query. Contains full information about checkout

getUpdates

Use this method to receive incoming updates using long polling (wiki). An Array of Update objects is returned.

Parameters	Type	Required	Description
offset	Integer	Optional	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 getUpdates is called with an *offset* higher than its *update_id*. The negative offset can be specified to retrieve updates starting from *-offset* update from the end of the updates queue. All previous updates will forgotten.
limit	Integer	Optional	Limits the number of updates to be retrieved. Values between 1—100 are accepted. Defaults to 100.
timeout	Integer	Optional	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	Array of String	Optional	List the types of updates you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used. Please note that this parameter doesn't affect updates created before the call to the getUpdates, so unwanted updates may be received for a short period of time.

> 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.

setWebhook

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

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

Parameters	Type	Required	Description
url	String	Yes	HTTPS url to send updates to. Use an empty string to remove webhook integration
certificate	InputFile	Optional	Upload your public key certificate so that the root certificate in use can be checked. See our self-signed guide for details.
max_connections	Integer	Optional	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	Array of String	Optional	List the types of updates you want your bot to receive. For example, specify [“message”, “edited_channel_post”, “callback_query”] to only receive updates of these types. See Update for a complete list of available update types. Specify an empty list to receive all updates regardless of type (default). If not specified, the previous setting will be used. Please note that this parameter doesn't affect updates created before the call to the setWebhook, so unwanted updates may be received for a short period of time.

> Notes
> 1. You will not be able to receive updates using 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.
>
> NEW! If you're having any trouble setting up webhooks, please check out this amazing guide to Webhooks.

deleteWebhook

Use this method to remove webhook integration if you decide to switch back to getUpdates. Returns *True* on success. Requires no parameters.

getWebhookInfo

Use this method to get current webhook status. Requires no parameters. On success, returns a WebhookInfo object. If the bot is using getUpdates, will return an object with the *url* field empty.

WebhookInfo

Contains information about the current status of a webhook.

Field	Type	Description
url	String	Webhook URL, may be empty if webhook is not set up
has_custom_certificate	Boolean	True, if a custom certificate was provided for webhook certificate checks
pending_update_count	Integer	Number of updates awaiting delivery
last_error_date	Integer	*Optional*. Unix time for the most recent error that happened when trying to deliver an update via webhook
last_error_message	String	*Optional*. Error message in human-readable format for the most recent error that happened when trying to deliver an update via webhook
max_connections	Integer	*Optional*. Maximum allowed number of simultaneous HTTPS connections to the webhook for update delivery
allowed_updates	Array of String	*Optional*. A list of update types the bot is subscribed to. Defaults to all update types

Available types

All types used in the Bot API responses are represented as JSON-objects.

It is safe to use 32-bit signed integers for storing all **Integer** fields unless otherwise noted.

> **Optional** fields may be not returned when irrelevant.

User

This object represents a Telegram user or bot.

Field	Type	Description
id	Integer	Unique identifier for this user or bot
is_bot	Boolean	True, if this user is a bot
first_name	String	User‘s or bot’s first name
last_name	String	*Optional*. User‘s or bot’s last name
username	String	*Optional*. User‘s or bot’s username
language_code	String	*Optional*. IETF language tag of the user's language

Chat

This object represents a chat.

Field	Type	Description
id	Integer	Unique identifier for this chat. This number 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.
type	String	Type of chat, can be either “private”, “group”, “supergroup” or “channel”
title	String	*Optional*. Title, for supergroups, channels and group chats
username	String	*Optional*. Username, for private chats, supergroups and channels if available
first_name	String	*Optional*. First name of the other party in a private chat
last_name	String	*Optional*. Last name of the other party in a private chat
all_members_are_administrators	Boolean	*Optional*. True if a group has ‘All Members Are Admins’ enabled.
photo	ChatPhoto	*Optional*. Chat photo. Returned only in getChat.
description	String	*Optional*. Description, for supergroups and channel chats. Returned only in getChat.
invite_link	String	*Optional*. Chat invite link, for supergroups and channel chats. Returned only in getChat.
pinned_message	Message	*Optional*. Pinned message, for supergroups and channel chats. Returned only in getChat.
sticker_set_name	String	*Optional*. For supergroups, name of group sticker set. Returned only in getChat.
can_set_sticker_set	Boolean	*Optional*. True, if the bot can change the group sticker set. Returned only in getChat.

Message

This object represents a message.

Field	Type	Description
message_id	Integer	Unique message identifier inside this chat
from	User	*Optional*. Sender, empty for messages sent to channels
date	Integer	Date the message was sent in Unix time
chat	Chat	Conversation the message belongs to
forward_from	User	*Optional*. For forwarded messages, sender of the original message
forward_from_chat	Chat	*Optional*. For messages forwarded from channels, information about the original channel
forward_from_message_id	Integer	*Optional*. For messages forwarded from channels, identifier of the original message in the channel
forward_signature	String	*Optional*. For messages forwarded from channels, signature of the post author if present
forward_date	Integer	*Optional*. For forwarded messages, date the original message was sent in Unix time
reply_to_message	Message	*Optional*. For replies, the original message. Note that the Message object in this field will not contain further *reply_to_message* fields even if it itself is a reply.
edit_date	Integer	*Optional*. Date the message was last edited in Unix time
media_group_id	String	*Optional*. The unique identifier of a media message group this message belongs to
author_signature	String	*Optional*. Signature of the post author for messages in channels
text	String	*Optional*. For text messages, the actual UTF-8 text of the message, 0-4096 characters.
entities	Array of MessageEntity	*Optional*. For text messages, special entities like usernames, URLs, bot commands, etc. that appear in the text
caption_entities	Array of MessageEntity	*Optional*. For messages with a caption, special entities like usernames, URLs, bot commands, etc. that appear in the caption
audio	Audio	*Optional*. Message is an audio file, information about the file
document	Document	*Optional*. Message is a general file, information about the file
animation	Animation	*Optional*. Message is an animation, information about the animation. For backward compatibility, when this field is set, the *document* field will also be set
game	Game	*Optional*. Message is a game, information about the game. More about games »
photo	Array of PhotoSize	*Optional*. Message is a photo, available sizes of the photo
sticker	Sticker	*Optional*. Message is a sticker, information about the sticker

video	Video	*Optional*. Message is a video, information about the video
voice	Voice	*Optional*. Message is a voice message, information about the file
video_note	VideoNote	*Optional*. Message is a video note, information about the video message
caption	String	*Optional*. Caption for the audio, document, photo, video or voice, 0-1024 characters
contact	Contact	*Optional*. Message is a shared contact, information about the contact
location	Location	*Optional*. Message is a shared location, information about the location
venue	Venue	*Optional*. Message is a venue, information about the venue
new_chat_members	Array of User	*Optional*. New members that were added to the group or supergroup and information about them (the bot itself may be one of these members)
left_chat_member	User	*Optional*. A member was removed from the group, information about them (this member may be the bot itself)
new_chat_title	String	*Optional*. A chat title was changed to this value
new_chat_photo	Array of PhotoSize	*Optional*. A chat photo was change to this value
delete_chat_photo	True	*Optional*. Service message: the chat photo was deleted
group_chat_created	True	*Optional*. Service message: the group has been created
supergroup_chat_created	True	*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	True	*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.
migrate_to_chat_id	Integer	*Optional*. The group has been migrated to a supergroup with the specified identifier. This number 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.
migrate_from_chat_id	Integer	*Optional*. The supergroup has been migrated from a group with the specified identifier. This number 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.
pinned_message	Message	*Optional*. Specified message was pinned. Note that the Message object in this field will not contain further *reply_to_message* fields even if it is itself a reply.
invoice	Invoice	*Optional*. Message is an invoice for a payment, information about the invoice. More about payments »

successful_payment	SuccessfulPayment	*Optional*. Message is a service message about a successful payment, information about the payment. More about payments »
connected_website	String	*Optional*. The domain name of the website on which the user has logged in. More about Telegram Login »
passport_data	PassportData	*Optional*. Telegram Passport data

MessageEntity

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

Field	Type	Description
type	String	Type of the entity. Can be *mention* (`@username`), *hashtag*, *cashtag*, *bot_command*, *url*, *email*, *phone_number*, *bold* (bold text), *italic* (italic text), *code* (monowidth string), *pre* (monowidth block), *text_link* (for clickable text URLs), *text_mention* (for users without usernames)
offset	Integer	Offset in UTF-16 code units to the start of the entity
length	Integer	Length of the entity in UTF-16 code units
url	String	*Optional*. For “text_link” only, url that will be opened after user taps on the text
user	User	*Optional*. For “text_mention” only, the mentioned user

PhotoSize

This object represents one size of a photo or a file / sticker thumbnail.

Field	Type	Description
file_id	String	Unique identifier for this file
width	Integer	Photo width
height	Integer	Photo height
file_size	Integer	*Optional*. File size

Audio

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

Field	Type	Description
file_id	String	Unique identifier for this file
duration	Integer	Duration of the audio in seconds as defined by sender
performer	String	*Optional*. Performer of the audio as defined by sender or by audio tags
title	String	*Optional*. Title of the audio as defined by sender or by audio tags
mime_type	String	*Optional*. MIME type of the file as defined by sender
file_size	Integer	*Optional*. File size
thumb	PhotoSize	*Optional*. Thumbnail of the album cover to which the music file belongs

Document

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

Field	Type	Description
file_id	String	Unique file identifier
thumb	PhotoSize	*Optional*. Document thumbnail as defined by sender
file_name	String	*Optional*. Original filename as defined by sender
mime_type	String	*Optional*. MIME type of the file as defined by sender
file_size	Integer	*Optional*. File size

Video

This object represents a video file.

Field	Type	Description
file_id	String	Unique identifier for this file
width	Integer	Video width as defined by sender
height	Integer	Video height as defined by sender
duration	Integer	Duration of the video in seconds as defined by sender
thumb	PhotoSize	*Optional*. Video thumbnail
mime_type	String	*Optional*. Mime type of a file as defined by sender
file_size	Integer	*Optional*. File size

Animation

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

Field	Type	Description
file_id	String	Unique file identifier
width	Integer	Video width as defined by sender
height	Integer	Video height as defined by sender
duration	Integer	Duration of the video in seconds as defined by sender
thumb	PhotoSize	*Optional*. Animation thumbnail as defined by sender
file_name	String	*Optional*. Original animation filename as defined by sender
mime_type	String	*Optional*. MIME type of the file as defined by sender
file_size	Integer	*Optional*. File size

Voice

This object represents a voice note.

Field	Type	Description

file_id	String	Unique identifier for this file
duration	Integer	Duration of the audio in seconds as defined by sender
mime_type	String	*Optional*. MIME type of the file as defined by sender
file_size	Integer	*Optional*. File size

VideoNote

This object represents a video message (available in Telegram apps as of v.4.0).

Field	Type	Description
file_id	String	Unique identifier for this file
length	Integer	Video width and height (diameter of the video message) as defined by sender
duration	Integer	Duration of the video in seconds as defined by sender
thumb	PhotoSize	*Optional*. Video thumbnail
file_size	Integer	*Optional*. File size

Contact

This object represents a phone contact.

Field	Type	Description
phone_number	String	Contact's phone number
first_name	String	Contact's first name
last_name	String	*Optional*. Contact's last name
user_id	Integer	*Optional*. Contact's user identifier in Telegram
vcard	String	*Optional*. Additional data about the contact in the form of a vCard

Location

This object represents a point on the map.

Field	Type	Description
longitude	Float	Longitude as defined by sender
latitude	Float	Latitude as defined by sender

Venue

This object represents a venue.

Field	Type	Description
location	Location	Venue location
title	String	Name of the venue
address	String	Address of the venue

foursquare_id	String	*Optional*. Foursquare identifier of the venue
foursquare_type	String	*Optional*. Foursquare type of the venue. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)

UserProfilePhotos

This object represent a user's profile pictures.

Field	Type	Description
total_count	Integer	Total number of profile pictures the target user has
photos	Array of Array of PhotoSize	Requested profile pictures (in up to 4 sizes each)

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 getFile.

> Maximum file size to download is 20 MB

Field	Type	Description
file_id	String	Unique identifier for this file
file_size	Integer	*Optional*. File size, if known
file_path	String	*Optional*. File path. Use `https://api.telegram.org/file/bot<token>/<file_path>` to get the file.

ReplyKeyboardMarkup

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

Field	Type	Description
keyboard	Array of Array of KeyboardButton	Array of button rows, each represented by an Array of KeyboardButton objects
resize_keyboard	Boolean	*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	Boolean	*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*.
selective	Boolean	*Optional*. Use this parameter if you want to show the keyboard to specific users only. Targets: 1) users that are @mentioned in the *text* of the Message object; 2) if the bot's message is a reply (has *reply_to_message_id*), sender of the original message. *Example:* A user requests to change the bot‘s language, bot replies to the request with a keyboard to select the new language. Other users in the group don’t see the keyboard.

KeyboardButton

This object represents one button of the reply keyboard. For simple text buttons *String* can be used instead of this object to specify text of the button. Optional fields are mutually exclusive.

Field	Type	Description
text	String	Text of the button. If none of the optional fields are used, it will be sent as a message when the button is pressed
request_contact	Boolean	*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	Boolean	*Optional*. If *True*, the user's current location will be sent when the button is pressed. Available in private chats only

Note: *request_contact* and *request_location* options will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

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 ReplyKeyboardMarkup).

Field	Type	Description
remove_keyboard	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 ReplyKeyboardMarkup)
selective	Boolean	*Optional*. Use this parameter if you want to remove the keyboard for specific users only. Targets: 1) users that are @mentioned in the *text* of the Message object; 2) if the bot's message is a reply (has *reply_to_message_id*), sender of the original message. *Example:* A user votes in a poll, bot returns confirmation message in reply to the vote and removes the keyboard for that user, while still showing the keyboard with poll options to users who haven't voted yet.

InlineKeyboardMarkup

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

Field	Type	Description
inline_keyboard	Array of Array of InlineKeyboardButton	Array of button rows, each represented by an Array of InlineKeyboardButton objects

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

InlineKeyboardButton

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

Field	Type	Description
text	String	Label text on the button
url	String	*Optional*. HTTP or tg:// url to be opened when button is pressed
callback_data	String	*Optional*. Data to be sent in a callback query to the bot when button is pressed, 1-64 bytes

switch_inline_query	String	*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. Can be empty, in which case just the bot’s username will be inserted. Note: This offers an easy way for users to start using your bot in inline mode when they are currently in a private chat with it. Especially useful when combined with *switch_pm…* actions – in this case the user will be automatically returned to the chat they switched from, skipping the chat selection screen.
switch_inline_query_current_chat	String	*Optional*. If set, pressing the button will insert the bot‘s username and the specified inline query in the current chat's input field. Can be empty, in which case only the bot’s username will be inserted. This offers a quick way for the user to open your bot in inline mode in the same chat – good for selecting something from multiple options.
callback_game	CallbackGame	*Optional*. Description of the game that will be launched when the user presses the button. NOTE: This type of button **must** always be the first button in the first row.
pay	Boolean	*Optional*. Specify True, to send a Pay button. NOTE: This type of button **must** always be the first button in the first row.

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 *message* will be present. If the button was attached to a message sent via the bot (in inline mode), the field *inline_message_id* will be present. Exactly one of the fields *data* or *game_short_name* will be present.

Field	Type	Description
id	String	Unique identifier for this query
from	User	Sender
message	Message	*Optional*. Message with the callback button that originated the query. Note that message content and message date will not be available if the message is too old
inline_message_id	String	*Optional*. Identifier of the message sent via the bot in inline mode, that originated the query.
chat_instance	String	Global identifier, uniquely corresponding to the chat to which the message with the callback button was sent. Useful for high scores in games.
data	String	*Optional*. Data associated with the callback button. Be aware that a bad client can send arbitrary data in this field.
game_short_name	String	*Optional*. Short name of a Game to be returned, serves as the unique identifier for the game

> NOTE: After the user presses a callback button, Telegram clients will display a progress bar until you call answerCallbackQuery. It is, therefore, necessary to react by calling

> answerCallbackQuery even if no notification to the user is needed (e.g., without specifying any of the optional parameters).

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.

Field	Type	Description
force_reply	True	Shows reply interface to the user, as if they manually selected the bot‘s message and tapped ’Reply'
selective	Boolean	*Optional*. Use this parameter if you want to force reply from specific users only. Targets: 1) users that are @mentioned in the *text* of the Message object; 2) if the bot's message is a reply (has *reply_to_message_id*), sender of the original message.

> 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 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.

ChatPhoto

This object represents a chat photo.

Field	Type	Description
small_file_id	String	Unique file identifier of small (160x160) chat photo. This file_id can be used only for photo download.
big_file_id	String	Unique file identifier of big (640x640) chat photo. This file_id can be used only for photo download.

ChatMember

This object contains information about one member of a chat.

Field	Type	Description
user	User	Information about the user
status	String	The member's status in the chat. Can be “creator”, “administrator”, “member”, “restricted”, “left” or “kicked”
until_date	Integer	*Optional*. Restricted and kicked only. Date when restrictions will be lifted for this user, unix time
can_be_edited	Boolean	*Optional*. Administrators only. True, if the bot is allowed to edit administrator privileges of that user
can_change_info	Boolean	*Optional*. Administrators only. True, if the administrator can change the chat title, photo and other settings

can_post_messages	Boolean	*Optional*. Administrators only. True, if the administrator can post in the channel, channels only
can_edit_messages	Boolean	*Optional*. Administrators only. True, if the administrator can edit messages of other users and can pin messages, channels only
can_delete_messages	Boolean	*Optional*. Administrators only. True, if the administrator can delete messages of other users
can_invite_users	Boolean	*Optional*. Administrators only. True, if the administrator can invite new users to the chat
can_restrict_members	Boolean	*Optional*. Administrators only. True, if the administrator can restrict, ban or unban chat members
can_pin_messages	Boolean	*Optional*. Administrators only. True, if the administrator can pin messages, supergroups only
can_promote_members	Boolean	*Optional*. Administrators only. True, if the administrator can add new administrators with a subset of his own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by the user)
can_send_messages	Boolean	*Optional*. Restricted only. True, if the user can send text messages, contacts, locations and venues
can_send_media_messages	Boolean	*Optional*. Restricted only. True, if the user can send audios, documents, photos, videos, video notes and voice notes, implies can_send_messages
can_send_other_messages	Boolean	*Optional*. Restricted only. True, if the user can send animations, games, stickers and use inline bots, implies can_send_media_messages
can_add_web_page_previews	Boolean	*Optional*. Restricted only. True, if user may add web page previews to his messages, implies can_send_media_messages

ResponseParameters

Contains information about why a request was unsuccessful.

Field	Type	Description
migrate_to_chat_id	Integer	*Optional*. The group has been migrated to a supergroup with the specified identifier. This number 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.
retry_after	Integer	*Optional*. In case of exceeding flood control, the number of seconds left to wait before the request can be repeated

InputMedia

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

- InputMediaAnimation
- InputMediaDocument
- InputMediaAudio
- InputMediaPhoto
- InputMediaVideo

InputMediaPhoto

Represents a photo to be sent.

Field	Type	Description
type	String	Type of the result, must be *photo*
media	String	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 info on Sending Files »
caption	String	*Optional*. Caption of the photo to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.

InputMediaVideo

Represents a video to be sent.

Field	Type	Description
type	String	Type of the result, must be *video*
media	String	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 info on Sending Files »
thumb	InputFile or String	*Optional*. Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail‘s width and height should not exceed 90. 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 info on Sending Files »
caption	String	*Optional*. Caption of the video to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
width	Integer	*Optional*. Video width
height	Integer	*Optional*. Video height
duration	Integer	*Optional*. Video duration
supports_streaming	Boolean	*Optional*. Pass *True*, if the uploaded video is suitable for streaming

InputMediaAnimation

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

Field	Type	Description
type	String	Type of the result, must be *animation*
media	String	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 info on Sending Files »
thumb	InputFile or String	*Optional*. Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail‘s width and height should not exceed 90. 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 info on Sending Files »
caption	String	*Optional*. Caption of the animation to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
width	Integer	*Optional*. Animation width
height	Integer	*Optional*. Animation height
duration	Integer	*Optional*. Animation duration

InputMediaAudio

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

Field	Type	Description
type	String	Type of the result, must be *audio*
media	String	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 info on Sending Files »
thumb	InputFile or String	*Optional*. Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 90. 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 info on Sending Files »
caption	String	*Optional*. Caption of the audio to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
duration	Integer	*Optional*. Duration of the audio in seconds
performer	String	*Optional*. Performer of the audio
title	String	*Optional*. Title of the audio

InputMediaDocument

Represents a general file to be sent.

Field	Type	Description
type	String	Type of the result, must be *document*
media	String	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 info on Sending Files »
thumb	InputFile or String	*Optional*. Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 90. 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 info on Sending Files »
caption	String	*Optional*. Caption of the document to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.

InputFile

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.

Sending files

There are three ways to send files (photos, stickers, audio, media, etc.):

1. If the file is already stored somewhere on the Telegram servers, you don't need to reupload it: each file object has a **file_id** field, simply pass this **file_id** as a parameter instead of uploading. There are **no limits** for files sent this way.
2. Provide Telegram with an HTTP URL for the file to be sent. Telegram will download and send the file. 5 MB max size for photos and 20 MB max for other types of content.
3. Post the file using multipart/form-data in the usual way that files are uploaded via the browser. 10 MB max size for photos, 50 MB for other files.

Sending by file_id

- It is not possible to change the file type when resending by **file_id**. I.e. a video can't be sent as a photo, a photo can't be sent as a document, etc.
- It is not possible to resend thumbnails.
- Resending a photo by **file_id** will send all of its sizes.
- **file_id** is unique for each individual bot and can't be transferred from one bot to another.

Sending by URL

- When sending by URL the target file must have the correct MIME type (e.g., audio/mpeg for sendAudio, etc.).
- In sendDocument, sending by URL will currently only work for **gif**, **pdf** and **zip** files.
- To use sendVoice, the file must have the type audio/ogg and be no more than 1MB in size. 1-20MB voice notes will be sent as files.
- Other configurations may work but we can't guarantee that they will.

Inline mode objects

Objects and methods used in the inline mode are described in the Inline mode section.

Available methods

> All methods in the Bot API are case-insensitive. We support **GET** and **POST** HTTP methods.
> Use either URL query string or *application/json* or *application/x-www-form-urlencoded* or *multipart/form-data* for passing parameters in Bot API requests.
> On successful call, a JSON-object containing the result will be returned.

getMe

A simple method for testing your bot's auth token. Requires no parameters. Returns basic information about the bot in form of a User object.

sendMessage

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)

text	String	Yes	Text of the message to be sent
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
disable_web_page_preview	Boolean	Optional	Disables link previews for links in this message
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

Formatting options

The Bot API supports basic formatting for messages. You can use bold and italic text, as well as inline links and pre-formatted code in your bots' messages. Telegram clients will render them accordingly. You can use either markdown-style or HTML-style formatting.

Note that Telegram clients will display an **alert** to the user before opening an inline link ('Open this link?' together with the full URL).

Links `tg://user?id=<user_id>` can be used to mention a user by their id without using a username. Please note:

- These links will work **only** if they are used inside an inline link. For example, they will not work, when used in an inline keyboard button or in a message text.
- These mentions are only guaranteed to work if the user has contacted the bot in the past, has sent a callback query to the bot via inline button or is a member in the group where he was mentioned.

Markdown style

To use this mode, pass *Markdown* in the *parse_mode* field when using sendMessage. Use the following syntax in your message:

```
*bold text*
_italic text_
[inline URL](http://www.example.com/)
[inline mention of a user](tg://user?id=123456789)
`inline fixed-width code`
```block_language
pre-formatted fixed-width code block
```
```

HTML style

To use this mode, pass *HTML* in the *parse_mode* field when using sendMessage. The following tags are currently supported:

```
<b>bold</b>, <strong>bold</strong>
<i>italic</i>, <em>italic</em>
<a href="http://www.example.com/">inline URL</a>
<a href="tg://user?id=123456789">inline mention of a user</a>
<code>inline fixed-width code</code>
<pre>pre-formatted fixed-width code block</pre>
```

Please note:

- Only the tags mentioned above are currently supported.
- Tags must not be nested.

- All `<`, `>` and `&` symbols that are not a part of a tag or an HTML entity must be replaced with the corresponding HTML entities (`<` with `<`, `>` with `>` and `&` with `&`).
- All numerical HTML entities are supported.
- The API currently supports only the following named HTML entities: `<`, `>`, `&` and `&quot;`.

forwardMessage

Use this method to forward messages of any kind. On success, the sent Message is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
from_chat_id	Integer or String	Yes	Unique identifier for the chat where the original message was sent (or channel username in the format `@channelusername`)
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
message_id	Integer	Yes	Message identifier in the chat specified in *from_chat_id*

sendPhoto

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
photo	InputFile or String	Yes	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. More info on Sending Files »
caption	String	Optional	Photo caption (may also be used when resending photos by *file_id*), 0-1024 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendAudio

Use this method to send audio files, if you want Telegram clients to display them in the music player. Your audio must be in the .mp3 format. On success, the sent Message is returned. Bots can currently send audio files of up to 50 MB in size, this limit may be changed in the future.

For sending voice messages, use the sendVoice method instead.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
audio	InputFile or String	Yes	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 info on Sending Files »
caption	String	Optional	Audio caption, 0-1024 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
duration	Integer	Optional	Duration of the audio in seconds
performer	String	Optional	Performer
title	String	Optional	Track name
thumb	InputFile or String	Optional	Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail‘s width and height should not exceed 90. 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 info on Sending Files »
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendDocument

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
document	InputFile or String	Yes	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 info on Sending Files »
thumb	InputFile or String	Optional	Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail‘s width

			and height should not exceed 90. 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 info on Sending Files »
caption	String	Optional	Document caption (may also be used when resending documents by *file_id*), 0-1024 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendVideo

Use this method to send video files, Telegram clients support mp4 videos (other formats may be sent as Document). On success, the sent Message is returned. Bots can currently send video files of up to 50 MB in size, this limit may be changed in the future.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
video	InputFile or String	Yes	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 info on Sending Files »
duration	Integer	Optional	Duration of sent video in seconds
width	Integer	Optional	Video width
height	Integer	Optional	Video height
thumb	InputFile or String	Optional	Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail's width and height should not exceed 90. 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 info on Sending Files »
caption	String	Optional	Video caption (may also be used when resending videos by *file_id*), 0-1024 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to

			show bold, italic, fixed-width text or inline URLs in the media caption.
supports_streaming	Boolean	Optional	Pass *True*, if the uploaded video is suitable for streaming
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendAnimation

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
animation	InputFile or String	Yes	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 info on Sending Files »
duration	Integer	Optional	Duration of sent animation in seconds
width	Integer	Optional	Animation width
height	Integer	Optional	Animation height
thumb	InputFile or String	Optional	Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail‘s width and height should not exceed 90. 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 info on Sending Files »
caption	String	Optional	Animation caption (may also be used when resending animation by *file_id*), 0-1024 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup	Optional	Additional interface options. A JSON-serialized object for an

	or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply		inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendVoice

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
voice	InputFile or String	Yes	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 info on Sending Files »
caption	String	Optional	Voice message caption, 0-1024 characters
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
duration	Integer	Optional	Duration of the voice message in seconds
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendVideoNote

As of v.4.0, Telegram clients support rounded square mp4 videos of up to 1 minute long. Use this method to send video messages. On success, the sent Message is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
video_note	InputFile or String	Yes	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 info on Sending Files ». Sending video notes by a URL is currently unsupported
duration	Integer	Optional	Duration of sent video in seconds

length	Integer	Optional	Video width and height, i.e. diameter of the video message
thumb	InputFile or String	Optional	Thumbnail of the file sent. The thumbnail should be in JPEG format and less than 200 kB in size. A thumbnail‘s width and height should not exceed 90. 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 info on Sending Files »
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendMediaGroup

Use this method to send a group of photos or videos as an album. On success, an array of the sent Messages is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
media	Array of InputMediaPhoto and InputMediaVideo	Yes	A JSON-serialized array describing photos and videos to be sent, must include 2–10 items
disable_notification	Boolean	Optional	Sends the messages silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the messages are a reply, ID of the original message

sendLocation

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
latitude	Float number	Yes	Latitude of the location
longitude	Float number	Yes	Longitude of the location
live_period	Integer	Optional	Period in seconds for which the location will be updated (see Live Locations, should be between 60 and 86400.
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message

reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

editMessageLiveLocation

Use this method to edit live location messages sent by the bot or via the bot (for inline bots). A location can be edited until its *live_period* expires or editing is explicitly disabled by a call to stopMessageLiveLocation. On success, if the edited message was sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	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	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
latitude	Float number	Yes	Latitude of new location
longitude	Float number	Yes	Longitude of new location
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for a new inline keyboard.

stopMessageLiveLocation

Use this method to stop updating a live location message sent by the bot or via the bot (for inline bots) before *live_period* expires. On success, if the message was sent by the bot, the sent Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	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	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for a new inline keyboard.

sendVenue

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
latitude	Float number	Yes	Latitude of the venue
longitude	Float number	Yes	Longitude of the venue

title	String	Yes	Name of the venue
address	String	Yes	Address of the venue
foursquare_id	String	Optional	Foursquare identifier of the venue
foursquare_type	String	Optional	Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

sendContact

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
phone_number	String	Yes	Contact's phone number
first_name	String	Yes	Contact's first name
last_name	String	Optional	Contact's last name
vcard	String	Optional	Additional data about the contact in the form of a vCard, 0-2048 bytes
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove keyboard or to force a reply from the user.

sendChatAction

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 sendChatAction with *action* = *upload_photo*. The user will see a “sending photo” status for the bot.

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

Parameters	Type	Required	Description

chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
action	String	Yes	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_audio* or *upload_audio* for audio files, *upload_document* for general files, *find_location* for location data, *record_video_note* or *upload_video_note* for video notes.

getUserProfilePhotos

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

Parameters	Type	Required	Description
user_id	Integer	Yes	Unique identifier of the target user
offset	Integer	Optional	Sequential number of the first photo to be returned. By default, all photos are returned.
limit	Integer	Optional	Limits the number of photos to be retrieved. Values between 1—100 are accepted. Defaults to 100.

getFile

Use this method to get basic info about a file and prepare it for downloading. For the moment, bots can download files of up to 20MB in size. On success, a File object is returned. The file can then be downloaded via the link `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 getFile again.

Parameters	Type	Required	Description
file_id	String	Yes	File identifier to get info about

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

kickChatMember

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

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target group or username of the target supergroup or channel (in the format `@channelusername`)
user_id	Integer	Yes	Unique identifier of the target user
until_date	Integer	Optional	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

unbanChatMember

Use this method to unban a previously kicked user in a supergroup or channel. The user will **not** return to the group or channel automatically, but will be able to join via link, etc. The bot must be an administrator for this to work. Returns *True* on success.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target group or username of the target supergroup or channel (in the format `@username`)
user_id	Integer	Yes	Unique identifier of the target user

restrictChatMember

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup (in the format `@supergroupusername`)
user_id	Integer	Yes	Unique identifier of the target user
until_date	Integer	Optional	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
can_send_messages	Boolean	Optional	Pass True, if the user can send text messages, contacts, locations and venues
can_send_media_messages	Boolean	Optional	Pass True, if the user can send audios, documents, photos, videos, video notes and voice notes, implies can_send_messages
can_send_other_messages	Boolean	Optional	Pass True, if the user can send animations, games, stickers and use inline bots, implies can_send_media_messages
can_add_web_page_previews	Boolean	Optional	Pass True, if the user may add web page previews to their messages, implies can_send_media_messages

promoteChatMember

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
user_id	Integer	Yes	Unique identifier of the target user
can_change_info	Boolean	Optional	Pass True, if the administrator can change chat title, photo and other settings
can_post_messages	Boolean	Optional	Pass True, if the administrator can create channel posts, channels only

can_edit_messages	Boolean	Optional	Pass True, if the administrator can edit messages of other users and can pin messages, channels only
can_delete_messages	Boolean	Optional	Pass True, if the administrator can delete messages of other users
can_invite_users	Boolean	Optional	Pass True, if the administrator can invite new users to the chat
can_restrict_members	Boolean	Optional	Pass True, if the administrator can restrict, ban or unban chat members
can_pin_messages	Boolean	Optional	Pass True, if the administrator can pin messages, supergroups only
can_promote_members	Boolean	Optional	Pass True, if the administrator can add new administrators with a subset of his own privileges or demote administrators that he has promoted, directly or indirectly (promoted by administrators that were appointed by him)

exportChatInviteLink

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)

setChatPhoto

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

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
photo	InputFile	Yes	New chat photo, uploaded using multipart/form-data

deleteChatPhoto

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

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)

setChatTitle

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

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
title	String	Yes	New chat title, 1-255 characters

setChatDescription

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
description	String	Optional	New chat description, 0-255 characters

pinChatMessage

Use this method to pin a message in a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the 'can_pin_messages' admin right in the supergroup or 'can_edit_messages' admin right in the channel. Returns *True* on success.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
message_id	Integer	Yes	Identifier of a message to pin
disable_notification	Boolean	Optional	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.

unpinChatMessage

Use this method to unpin a message in a supergroup or a channel. The bot must be an administrator in the chat for this to work and must have the 'can_pin_messages' admin right in the supergroup or 'can_edit_messages' admin right in the channel. Returns *True* on success.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)

leaveChat

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup or channel (in the format `@channelusername`)

getChat

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup or channel (in the format `@channelusername`)

getChatAdministrators

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup or channel (in the format `@channelusername`)

getChatMembersCount

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

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup or channel (in the format `@channelusername`)

getChatMember

Use this method to get information about a member of a chat. Returns a ChatMember object on success.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup or channel (in the format `@channelusername`)
user_id	Integer	Yes	Unique identifier of the target user

setChatStickerSet

Use this method to set a new group sticker set for a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Use the field *can_set_sticker_set* optionally returned in getChat requests to check if the bot can use this method. Returns *True* on success.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup (in the format `@supergroupusername`)
sticker_set_name	String	Yes	Name of the sticker set to be set as the group sticker set

deleteChatStickerSet

Use this method to delete a group sticker set from a supergroup. The bot must be an administrator in the chat for this to work and must have the appropriate admin rights. Use the field *can_set_sticker_set* optionally returned in getChat requests to check if the bot can use this method. Returns *True* on success.

Parameters	Type	Required	Description

chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target supergroup (in the format `@supergroupusername`)

answerCallbackQuery

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.

Parameters	Type	Required	Description
callback_query_id	String	Yes	Unique identifier for the query to be answered
text	String	Optional	Text of the notification. If not specified, nothing will be shown to the user, 0-200 characters
show_alert	Boolean	Optional	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	String	Optional	URL that will be opened by the user's client. If you have created a 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 *callback_game* button. Otherwise, you may use links like `t.me/your_bot?start=XXXX` that open your bot with a parameter.
cache_time	Integer	Optional	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.

Inline mode methods

Methods and objects used in the inline mode are described in the Inline mode section.

Updating messages

The following methods allow you to change an existing message in the message history instead of sending a new one with a result of an action. This is most useful for messages with inline keyboards using callback queries, but can also help reduce clutter in conversations with regular chat bots.

Please note, that it is currently only possible to edit messages without *reply_markup* or with inline keyboards.

editMessageText

Use this method to edit text and game messages sent by the bot or via the bot (for inline bots). On success, if edited message is sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	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	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message

text	String	Yes	New text of the message
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
disable_web_page_preview	Boolean	Optional	Disables link previews for links in this message
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for an inline keyboard.

editMessageCaption

Use this method to edit captions of messages sent by the bot or via the bot (for inline bots). On success, if edited message is sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	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	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
caption	String	Optional	New caption of the message
parse_mode	String	Optional	Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for an inline keyboard.

editMessageMedia

Use this method to edit audio, document, photo, or video messages. If a message is a part of a message album, then it can be edited only to a photo or a video. Otherwise, message type can be changed arbitrarily. When inline message is edited, new file can't be uploaded. Use previously uploaded file via its file_id or specify a URL. On success, if the edited message was sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	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	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
media	InputMedia	Yes	A JSON-serialized object for a new media content of the message
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for a new inline keyboard.

editMessageReplyMarkup

Use this method to edit only the reply markup of messages sent by the bot or via the bot (for inline bots). On success, if edited message is sent by the bot, the edited Message is returned, otherwise *True* is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Optional	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	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message
reply_markup	InlineKeyboardMarkup	Optional	A JSON-serialized object for an inline keyboard.

deleteMessage

Use this method to delete a message, including service messages, with the following limitations:
- A message can only be deleted if it was sent less than 48 hours ago.
- Bots can delete outgoing messages in groups and supergroups.
- 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.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
message_id	Integer	Yes	Identifier of the message to delete

Stickers

The following methods and objects allow your bot to handle stickers and sticker sets.

Sticker

This object represents a sticker.

Field	Type	Description
file_id	String	Unique identifier for this file
width	Integer	Sticker width
height	Integer	Sticker height
thumb	PhotoSize	*Optional*. Sticker thumbnail in the .webp or .jpg format
emoji	String	*Optional*. Emoji associated with the sticker
set_name	String	*Optional*. Name of the sticker set to which the sticker belongs
mask_position	MaskPosition	*Optional*. For mask stickers, the position where the mask should be placed
file_size	Integer	*Optional*. File size

StickerSet

This object represents a sticker set.

Field	Type	Description
name	String	Sticker set name
title	String	Sticker set title
contains_masks	Boolean	*True*, if the sticker set contains masks
stickers	Array of Sticker	List of all set stickers

MaskPosition

This object describes the position on faces where a mask should be placed by default.

Field	Type	Description
point	String	The part of the face relative to which the mask should be placed. One of “forehead”, “eyes”, “mouth”, or “chin”.
x_shift	Float number	Shift by X-axis measured in widths of the mask scaled to the face size, from left to right. For example, choosing -1.0 will place mask just to the left of the default mask position.
y_shift	Float number	Shift by Y-axis measured in heights of the mask scaled to the face size, from top to bottom. For example, 1.0 will place the mask just below the default mask position.
scale	Float number	Mask scaling coefficient. For example, 2.0 means double size.

sendSticker

Use this method to send .webp stickers. On success, the sent Message is returned.

Parameters	Type	Required	Description
chat_id	Integer or String	Yes	Unique identifier for the target chat or username of the target channel (in the format `@channelusername`)
sticker	InputFile or String	Yes	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 file from the Internet, or upload a new one using multipart/form-data. More info on Sending Files »
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup or ReplyKeyboardMarkup or ReplyKeyboardRemove or ForceReply	Optional	Additional interface options. A JSON-serialized object for an inline keyboard, custom reply keyboard, instructions to remove reply keyboard or to force a reply from the user.

getStickerSet

Use this method to get a sticker set. On success, a StickerSet object is returned.

Parameters	Type	Required	Description

name	String	Yes	Name of the sticker set

uploadStickerFile

Use this method to upload a .png file with a sticker for later use in *createNewStickerSet* and *addStickerToSet* methods (can be used multiple times). Returns the uploaded File on success.

Parameters	Type	Required	Description
user_id	Integer	Yes	User identifier of sticker file owner
png_sticker	InputFile	Yes	**Png** image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. More info on Sending Files »

createNewStickerSet

Use this method to create new sticker set owned by a user. The bot will be able to edit the created sticker set. Returns *True* on success.

Parameters	Type	Required	Description
user_id	Integer	Yes	User identifier of created sticker set owner
name	String	Yes	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	String	Yes	Sticker set title, 1-64 characters
png_sticker	InputFile or String	Yes	**Png** image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a *file_id* as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the Internet, or upload a new one using multipart/form-data. More info on Sending Files »
emojis	String	Yes	One or more emoji corresponding to the sticker
contains_masks	Boolean	Optional	Pass *True*, if a set of mask stickers should be created
mask_position	MaskPosition	Optional	A JSON-serialized object for position where the mask should be placed on faces

addStickerToSet

Use this method to add a new sticker to a set created by the bot. Returns *True* on success.

Parameters	Type	Required	Description
user_id	Integer	Yes	User identifier of sticker set owner
name	String	Yes	Sticker set name
png_sticker	InputFile or String	Yes	**Png** image with the sticker, must be up to 512 kilobytes in size, dimensions must not exceed 512px, and either width or height must be exactly 512px. Pass a *file_id* as a String to send a file that already exists on the Telegram servers, pass an HTTP URL as a String for Telegram to get a file from the

			Internet, or upload a new one using multipart/form-data. More info on Sending Files »
emojis	String	Yes	One or more emoji corresponding to the sticker
mask_position	MaskPosition	Optional	A JSON-serialized object for position where the mask should be placed on faces

setStickerPositionInSet

Use this method to move a sticker in a set created by the bot to a specific position . Returns *True* on success.

Parameters	Type	Required	Description
sticker	String	Yes	File identifier of the sticker
position	Integer	Yes	New sticker position in the set, zero-based

deleteStickerFromSet

Use this method to delete a sticker from a set created by the bot. Returns *True* on success.

Parameters	Type	Required	Description
sticker	String	Yes	File identifier of the sticker

Inline mode

The following methods and objects allow your bot to work in inline mode.
Please see our Introduction to Inline bots for more details.

To enable this option, send the `/setinline` command to @BotFather and provide the placeholder text that the user will see in the input field after typing your bot's name.

InlineQuery

This object represents an incoming inline query. When the user sends an empty query, your bot could return some default or trending results.

Field	Type	Description
id	String	Unique identifier for this query
from	User	Sender
location	Location	*Optional*. Sender location, only for bots that request user location
query	String	Text of the query (up to 512 characters)
offset	String	Offset of the results to be returned, can be controlled by the bot

answerInlineQuery

Use this method to send answers to an inline query. On success, *True* is returned.
No more than 50 results per query are allowed.

Parameters	Type	Required	Description
inline_query_id	String	Yes	Unique identifier for the answered query
results	Array of InlineQueryResult	Yes	A JSON-serialized array of results for the inline query

cache_time	Integer	Optional	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	Boolean	Optional	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	String	Optional	Pass the offset that a client should send in the next query with the same text to receive more results. Pass an empty string if there are no more results or if you don‘t support pagination. Offset length can’t exceed 64 bytes.
switch_pm_text	String	Optional	If passed, clients will display a button with specified text that switches the user to a private chat with the bot and sends the bot a start message with the parameter *switch_pm_parameter*
switch_pm_parameter	String	Optional	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. *Example:* An inline bot that sends YouTube videos can ask the user to connect the bot to their YouTube account to adapt search results accordingly. To do this, it displays a ‘Connect your YouTube account’ button above the results, or even before showing any. The user presses the button, switches to a private chat with the bot and, in doing so, passes a start parameter that instructs the bot to return an oauth link. Once done, the bot can offer a *switch_inline* button so that the user can easily return to the chat where they wanted to use the bot's inline capabilities.

InlineQueryResult

This object represents one result of an inline query. Telegram clients currently support results of the following 20 types:

- InlineQueryResultCachedAudio
- InlineQueryResultCachedDocument
- InlineQueryResultCachedGif
- InlineQueryResultCachedMpeg4Gif
- InlineQueryResultCachedPhoto
- InlineQueryResultCachedSticker
- InlineQueryResultCachedVideo
- InlineQueryResultCachedVoice
- InlineQueryResultArticle
- InlineQueryResultAudio
- InlineQueryResultContact
- InlineQueryResultGame
- InlineQueryResultDocument
- InlineQueryResultGif
- InlineQueryResultLocation
- InlineQueryResultMpeg4Gif
- InlineQueryResultPhoto
- InlineQueryResultVenue
- InlineQueryResultVideo
- InlineQueryResultVoice

InlineQueryResultArticle

Represents a link to an article or web page.

Field	Type	Description
type	String	Type of the result, must be *article*

id	String	Unique identifier for this result, 1-64 Bytes
title	String	Title of the result
input_message_content	InputMessageContent	Content of the message to be sent
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
url	String	*Optional*. URL of the result
hide_url	Boolean	*Optional*. Pass *True*, if you don't want the URL to be shown in the message
description	String	*Optional*. Short description of the result
thumb_url	String	*Optional*. Url of the thumbnail for the result
thumb_width	Integer	*Optional*. Thumbnail width
thumb_height	Integer	*Optional*. Thumbnail height

InlineQueryResultPhoto

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.

Field	Type	Description
type	String	Type of the result, must be *photo*
id	String	Unique identifier for this result, 1-64 bytes
photo_url	String	A valid URL of the photo. Photo must be in **jpeg** format. Photo size must not exceed 5MB
thumb_url	String	URL of the thumbnail for the photo
photo_width	Integer	*Optional*. Width of the photo
photo_height	Integer	*Optional*. Height of the photo
title	String	*Optional*. Title for the result
description	String	*Optional*. Short description of the result
caption	String	*Optional*. Caption of the photo to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the photo

InlineQueryResultGif

Represents a link to an animated GIF file. By default, this animated GIF file will be sent by the user with optional caption. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the animation.

Field	Type	Description

type	String	Type of the result, must be *gif*
id	String	Unique identifier for this result, 1-64 bytes
gif_url	String	A valid URL for the GIF file. File size must not exceed 1MB
gif_width	Integer	*Optional*. Width of the GIF
gif_height	Integer	*Optional*. Height of the GIF
gif_duration	Integer	*Optional*. Duration of the GIF
thumb_url	String	URL of the static thumbnail for the result (jpeg or gif)
title	String	*Optional*. Title for the result
caption	String	*Optional*. Caption of the GIF file to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the GIF animation

InlineQueryResultMpeg4Gif

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.

Field	Type	Description
type	String	Type of the result, must be *mpeg4_gif*
id	String	Unique identifier for this result, 1-64 bytes
mpeg4_url	String	A valid URL for the MP4 file. File size must not exceed 1MB
mpeg4_width	Integer	*Optional*. Video width
mpeg4_height	Integer	*Optional*. Video height
mpeg4_duration	Integer	*Optional*. Video duration
thumb_url	String	URL of the static thumbnail (jpeg or gif) for the result
title	String	*Optional*. Title for the result
caption	String	*Optional*. Caption of the MPEG-4 file to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the video animation

InlineQueryResultVideo

Represents a link to a page containing an embedded video player or a video file. By default, this video file will be sent by the user with an optional caption. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the video.

> If an InlineQueryResultVideo message contains an embedded video (e.g., YouTube), you **must** replace its content using *input_message_content*.

Field	Type	Description
type	String	Type of the result, must be *video*
id	String	Unique identifier for this result, 1-64 bytes
video_url	String	A valid URL for the embedded video player or video file
mime_type	String	Mime type of the content of video url, “text/html” or “video/mp4”
thumb_url	String	URL of the thumbnail (jpeg only) for the video
title	String	Title for the result
caption	String	*Optional*. Caption of the video to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
video_width	Integer	*Optional*. Video width
video_height	Integer	*Optional*. Video height
video_duration	Integer	*Optional*. Video duration in seconds
description	String	*Optional*. Short description of the result
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the video. This field is **required** if InlineQueryResultVideo is used to send an HTML-page as a result (e.g., a YouTube video).

InlineQueryResultAudio

Represents a link to an mp3 audio file. By default, this audio file will be sent by the user. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the audio.

Field	Type	Description
type	String	Type of the result, must be *audio*
id	String	Unique identifier for this result, 1-64 bytes
audio_url	String	A valid URL for the audio file
title	String	Title
caption	String	*Optional*. Caption, 0-1024 characters

parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
performer	String	*Optional*. Performer
audio_duration	Integer	*Optional*. Audio duration in seconds
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the audio

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultVoice

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.

Field	Type	Description
type	String	Type of the result, must be *voice*
id	String	Unique identifier for this result, 1-64 bytes
voice_url	String	A valid URL for the voice recording
title	String	Recording title
caption	String	*Optional*. Caption, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
voice_duration	Integer	*Optional*. Recording duration in seconds
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the voice recording

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultDocument

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.

Field	Type	Description
type	String	Type of the result, must be *document*
id	String	Unique identifier for this result, 1-64 bytes
title	String	Title for the result
caption	String	*Optional*. Caption of the document to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to

		show bold, italic, fixed-width text or inline URLs in the media caption.
document_url	String	A valid URL for the file
mime_type	String	Mime type of the content of the file, either “application/pdf” or “application/zip”
description	String	*Optional*. Short description of the result
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the file
thumb_url	String	*Optional*. URL of the thumbnail (jpeg only) for the file
thumb_width	Integer	*Optional*. Thumbnail width
thumb_height	Integer	*Optional*. Thumbnail height

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultLocation

Represents a location on a map. By default, the location will be sent by the user. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the location.

Field	Type	Description
type	String	Type of the result, must be *location*
id	String	Unique identifier for this result, 1-64 Bytes
latitude	Float number	Location latitude in degrees
longitude	Float number	Location longitude in degrees
title	String	Location title
live_period	Integer	*Optional*. Period in seconds for which the location can be updated, should be between 60 and 86400.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the location
thumb_url	String	*Optional*. Url of the thumbnail for the result
thumb_width	Integer	*Optional*. Thumbnail width
thumb_height	Integer	*Optional*. Thumbnail height

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultVenue

Represents a venue. By default, the venue will be sent by the user. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the venue.

Field	Type	Description
type	String	Type of the result, must be *venue*

id	String	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	String	Title of the venue
address	String	Address of the venue
foursquare_id	String	*Optional*. Foursquare identifier of the venue if known
foursquare_type	String	*Optional*. Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the venue
thumb_url	String	*Optional*. Url of the thumbnail for the result
thumb_width	Integer	*Optional*. Thumbnail width
thumb_height	Integer	*Optional*. Thumbnail height

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultContact

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.

Field	Type	Description
type	String	Type of the result, must be *contact*
id	String	Unique identifier for this result, 1-64 Bytes
phone_number	String	Contact's phone number
first_name	String	Contact's first name
last_name	String	*Optional*. Contact's last name
vcard	String	*Optional*. Additional data about the contact in the form of a vCard, 0-2048 bytes
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the contact
thumb_url	String	*Optional*. Url of the thumbnail for the result
thumb_width	Integer	*Optional*. Thumbnail width
thumb_height	Integer	*Optional*. Thumbnail height

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultGame

Represents a Game.

Field	Type	Description
type	String	Type of the result, must be *game*
id	String	Unique identifier for this result, 1-64 bytes
game_short_name	String	Short name of the game
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message

Note: This will only work in Telegram versions released after October 1, 2016. Older clients will not display any inline results if a game result is among them.

InlineQueryResultCachedPhoto

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.

Field	Type	Description
type	String	Type of the result, must be *photo*
id	String	Unique identifier for this result, 1-64 bytes
photo_file_id	String	A valid file identifier of the photo
title	String	*Optional*. Title for the result
description	String	*Optional*. Short description of the result
caption	String	*Optional*. Caption of the photo to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the photo

InlineQueryResultCachedGif

Represents a link to an animated GIF file stored on the Telegram servers. By default, this animated GIF file will be sent by the user with an optional caption. Alternatively, you can use *input_message_content* to send a message with specified content instead of the animation.

Field	Type	Description
type	String	Type of the result, must be *gif*
id	String	Unique identifier for this result, 1-64 bytes
gif_file_id	String	A valid file identifier for the GIF file
title	String	*Optional*. Title for the result
caption	String	*Optional*. Caption of the GIF file to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to

		show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the GIF animation

InlineQueryResultCachedMpeg4Gif

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.

Field	Type	Description
type	String	Type of the result, must be *mpeg4_gif*
id	String	Unique identifier for this result, 1-64 bytes
mpeg4_file_id	String	A valid file identifier for the MP4 file
title	String	*Optional*. Title for the result
caption	String	*Optional*. Caption of the MPEG-4 file to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the video animation

InlineQueryResultCachedSticker

Represents a link to a sticker stored on the Telegram servers. By default, this sticker will be sent by the user. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the sticker.

Field	Type	Description
type	String	Type of the result, must be *sticker*
id	String	Unique identifier for this result, 1-64 bytes
sticker_file_id	String	A valid file identifier of the sticker
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the sticker

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultCachedDocument

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.

Field	Type	Description
type	String	Type of the result, must be *document*

id	String	Unique identifier for this result, 1-64 bytes
title	String	Title for the result
document_file_id	String	A valid file identifier for the file
description	String	*Optional*. Short description of the result
caption	String	*Optional*. Caption of the document to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the file

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultCachedVideo

Represents a link to a video file stored on the Telegram servers. By default, this video file will be sent by the user with an optional caption. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the video.

Field	Type	Description
type	String	Type of the result, must be *video*
id	String	Unique identifier for this result, 1-64 bytes
video_file_id	String	A valid file identifier for the video file
title	String	Title for the result
description	String	*Optional*. Short description of the result
caption	String	*Optional*. Caption of the video to be sent, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the video

InlineQueryResultCachedVoice

Represents a link to a voice message stored on the Telegram servers. By default, this voice message will be sent by the user. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the voice message.

Field	Type	Description
type	String	Type of the result, must be *voice*
id	String	Unique identifier for this result, 1-64 bytes
voice_file_id	String	A valid file identifier for the voice message

title	String	Voice message title
caption	String	*Optional*. Caption, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the voice message

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InlineQueryResultCachedAudio

Represents a link to an mp3 audio file stored on the Telegram servers. By default, this audio file will be sent by the user. Alternatively, you can use *input_message_content* to send a message with the specified content instead of the audio.

Field	Type	Description
type	String	Type of the result, must be *audio*
id	String	Unique identifier for this result, 1-64 bytes
audio_file_id	String	A valid file identifier for the audio file
caption	String	*Optional*. Caption, 0-1024 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in the media caption.
reply_markup	InlineKeyboardMarkup	*Optional*. Inline keyboard attached to the message
input_message_content	InputMessageContent	*Optional*. Content of the message to be sent instead of the audio

Note: This will only work in Telegram versions released after 9 April, 2016. Older clients will ignore them.

InputMessageContent

This object represents the content of a message to be sent as a result of an inline query. Telegram clients currently support the following 4 types:

- InputTextMessageContent
- InputLocationMessageContent
- InputVenueMessageContent
- InputContactMessageContent

InputTextMessageContent

Represents the content of a text message to be sent as the result of an inline query.

Field	Type	Description
message_text	String	Text of the message to be sent, 1-4096 characters
parse_mode	String	*Optional*. Send *Markdown* or *HTML*, if you want Telegram apps to show bold, italic, fixed-width text or inline URLs in your bot's message.
disable_web_page_preview	Boolean	*Optional*. Disables link previews for links in the sent message

InputLocationMessageContent

Represents the content of a location message to be sent as the result of an inline query.

Field	Type	Description
latitude	Float	Latitude of the location in degrees
longitude	Float	Longitude of the location in degrees
live_period	Integer	*Optional*. Period in seconds for which the location can be updated, should be between 60 and 86400.

InputVenueMessageContent

Represents the content of a venue message to be sent as the result of an inline query.

Field	Type	Description
latitude	Float	Latitude of the venue in degrees
longitude	Float	Longitude of the venue in degrees
title	String	Name of the venue
address	String	Address of the venue
foursquare_id	String	*Optional*. Foursquare identifier of the venue, if known
foursquare_type	String	*Optional*. Foursquare type of the venue, if known. (For example, “arts_entertainment/default”, “arts_entertainment/aquarium” or “food/icecream”.)

InputContactMessageContent

Represents the content of a contact message to be sent as the result of an inline query.

Field	Type	Description
phone_number	String	Contact's phone number
first_name	String	Contact's first name
last_name	String	*Optional*. Contact's last name
vcard	String	*Optional*. Additional data about the contact in the form of a vCard, 0-2048 bytes

ChosenInlineResult

Represents a result of an inline query that was chosen by the user and sent to their chat partner.

Field	Type	Description
result_id	String	The unique identifier for the result that was chosen
from	User	The user that chose the result
location	Location	*Optional*. Sender location, only for bots that require user location
inline_message_id	String	*Optional*. Identifier of the sent inline message. Available only if there is an inline keyboard attached to the message. Will be also received in callback queries and can be used to edit the message.

query	String	The query that was used to obtain the result

Note: It is necessary to enable inline feedback via @Botfather in order to receive these objects in updates.

Payments

Your bot can accept payments from Telegram users. Please see the introduction to payments for more details on the process and how to set up payments for your bot. Please note that users will need Telegram v.4.0 or higher to use payments (released on May 18, 2017).

sendInvoice

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

Parameters	Type	Required	Description
chat_id	Integer	Yes	Unique identifier for the target private chat
title	String	Yes	Product name, 1-32 characters
description	String	Yes	Product description, 1-255 characters
payload	String	Yes	Bot-defined invoice payload, 1-128 bytes. This will not be displayed to the user, use for your internal processes.
provider_token	String	Yes	Payments provider token, obtained via Botfather
start_parameter	String	Yes	Unique deep-linking parameter that can be used to generate this invoice when used as a start parameter
currency	String	Yes	Three-letter ISO 4217 currency code, see more on currencies
prices	Array of LabeledPrice	Yes	Price breakdown, a list of components (e.g. product price, tax, discount, delivery cost, delivery tax, bonus, etc.)
provider_data	String	Optional	JSON-encoded 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	String	Optional	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	Integer	Optional	Photo size
photo_width	Integer	Optional	Photo width
photo_height	Integer	Optional	Photo height
need_name	Boolean	Optional	Pass *True*, if you require the user's full name to complete the order
need_phone_number	Boolean	Optional	Pass *True*, if you require the user's phone number to complete the order
need_email	Boolean	Optional	Pass *True*, if you require the user's email

			address to complete the order
need_shipping_address	Boolean	Optional	Pass *True*, if you require the user's shipping address to complete the order
send_phone_number_to_provider	Boolean	Optional	Pass *True*, if user's phone number should be sent to provider
send_email_to_provider	Boolean	Optional	Pass *True*, if user's email address should be sent to provider
is_flexible	Boolean	Optional	Pass *True*, if the final price depends on the shipping method
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup	Optional	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.

answerShippingQuery

If you sent an invoice requesting a shipping address and the parameter *is_flexible* was specified, the Bot API will send an Update with a *shipping_query* field to the bot. Use this method to reply to shipping queries. On success, True is returned.

Parameters	Type	Required	Description
shipping_query_id	String	Yes	Unique identifier for the query to be answered
ok	Boolean	Yes	Specify True if delivery to the specified address is possible and False if there are any problems (for example, if delivery to the specified address is not possible)
shipping_options	Array of ShippingOption	Optional	Required if *ok* is True. A JSON-serialized array of available shipping options.
error_message	String	Optional	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.

answerPreCheckoutQuery

Once the user has confirmed their payment and shipping details, the Bot API sends the final confirmation in the form of an Update with the field *pre_checkout_query*. Use this method to respond to such pre-checkout queries. On success, True is returned. Note: The Bot API must receive an answer within 10 seconds after the pre-checkout query was sent.

Parameters	Type	Required	Description
pre_checkout_query_id	String	Yes	Unique identifier for the query to be answered
ok	Boolean	Yes	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	String	Optional	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.

LabeledPrice

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

Field	Type	Description
label	String	Portion label
amount	Integer	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).

Invoice

This object contains basic information about an invoice.

Field	Type	Description
title	String	Product name
description	String	Product description
start_parameter	String	Unique bot deep-linking parameter that can be used to generate this invoice
currency	String	Three-letter ISO 4217 currency code
total_amount	Integer	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).

ShippingAddress

This object represents a shipping address.

Field	Type	Description
country_code	String	ISO 3166-1 alpha-2 country code
state	String	State, if applicable
city	String	City
street_line1	String	First line for the address
street_line2	String	Second line for the address
post_code	String	Address post code

OrderInfo

This object represents information about an order.

Field	Type	Description
name	String	*Optional*. User name
phone_number	String	*Optional*. User's phone number

email	String	*Optional*. User email
shipping_address	ShippingAddress	*Optional*. User shipping address

ShippingOption

This object represents one shipping option.

Field	Type	Description
id	String	Shipping option identifier
title	String	Option title
prices	Array of LabeledPrice	List of price portions

SuccessfulPayment

This object contains basic information about a successful payment.

Field	Type	Description
currency	String	Three-letter ISO 4217 currency code
total_amount	Integer	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	String	Bot specified invoice payload
shipping_option_id	String	*Optional*. Identifier of the shipping option chosen by the user
order_info	OrderInfo	*Optional*. Order info provided by the user
telegram_payment_charge_id	String	Telegram payment identifier
provider_payment_charge_id	String	Provider payment identifier

ShippingQuery

This object contains information about an incoming shipping query.

Field	Type	Description
id	String	Unique query identifier
from	User	User who sent the query
invoice_payload	String	Bot specified invoice payload
shipping_address	ShippingAddress	User specified shipping address

PreCheckoutQuery

This object contains information about an incoming pre-checkout query.

Field	Type	Description
id	String	Unique query identifier

from	User	User who sent the query
currency	String	Three-letter ISO 4217 currency code
total_amount	Integer	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	String	Bot specified invoice payload
shipping_option_id	String	*Optional*. Identifier of the shipping option chosen by the user
order_info	OrderInfo	*Optional*. Order info provided by the user

Telegram Passport

Telegram Passport is a unified authorization method for services that require personal identification. Users can upload their documents once, then instantly share their data with services that require real-world ID (finance, ICOs, etc.). Please see the manual for details.

PassportData

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

Parameters	Type	Description
data	Array of 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

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.

Field	Type	Description
file_id	String	Unique identifier for this file
file_size	Integer	File size
file_date	Integer	Unix time when the file was uploaded

EncryptedPassportElement

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

Parameters	Type	Description
type	String	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”.
data	String	*Optional*. Base64-encoded encrypted Telegram Passport element data provided by the user, available for “personal_details”, “passport”, “driver_license”, “identity_card”, “internal_passport” and “address” types. Can be decrypted and verified using the accompanying EncryptedCredentials.
phone_number	String	*Optional*. User's verified phone number, available only for “phone_number” type

email	String	*Optional*. User's verified email address, available only for “email” type
files	Array of PassportFile	*Optional*. Array of encrypted files with documents provided by the user, available for “utility_bill”, “bank_statement”, “rental_agreement”, “passport_registration” and “temporary_registration” types. Files can be decrypted and verified using the accompanying EncryptedCredentials.
front_side	PassportFile	*Optional*. Encrypted file with the front side of the document, provided by the user. Available for “passport”, “driver_license”, “identity_card” and “internal_passport”. The file can be decrypted and verified using the accompanying EncryptedCredentials.
reverse_side	PassportFile	*Optional*. Encrypted file with the reverse side of the document, provided by the user. Available for “driver_license” and “identity_card”. The file can be decrypted and verified using the accompanying EncryptedCredentials.
selfie	PassportFile	*Optional*. Encrypted file with the selfie of the user holding a document, provided by the user; available for “passport”, “driver_license”, “identity_card” and “internal_passport”. The file can be decrypted and verified using the accompanying EncryptedCredentials.
translation	Array of PassportFile	*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 EncryptedCredentials.
hash	String	Base64-encoded element hash for using in PassportElementErrorUnspecified

EncryptedCredentials

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

Parameters	Type	Description
data	String	Base64-encoded encrypted JSON-serialized data with unique user's payload, data hashes and secrets required for EncryptedPassportElement decryption and authentication
hash	String	Base64-encoded data hash for data authentication
secret	String	Base64-encoded secret, encrypted with the bot's public RSA key, required for data decryption

setPassportDataErrors

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.

Parameters	Type	Required	Description
user_id	Integer	Yes	User identifier
errors	Array of PassportElementError	Yes	A JSON-serialized array describing the errors

PassportElementError

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:

- PassportElementErrorDataField
- PassportElementErrorFrontSide
- PassportElementErrorReverseSide
- PassportElementErrorSelfie
- PassportElementErrorFile
- PassportElementErrorFiles
- PassportElementErrorTranslationFile
- PassportElementErrorTranslationFiles
- PassportElementErrorUnspecified

PassportElementErrorDataField

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.

Field	Type	Description
source	String	Error source, must be *data*
type	String	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	String	Name of the data field which has the error
data_hash	String	Base64-encoded data hash
message	String	Error message

PassportElementErrorFrontSide

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.

Field	Type	Description
source	String	Error source, must be *front_side*
type	String	The section of the user's Telegram Passport which has the issue, one of “passport”, “driver_license”, “identity_card”, “internal_passport”
file_hash	String	Base64-encoded hash of the file with the front side of the document
message	String	Error message

PassportElementErrorReverseSide

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.

Field	Type	Description
source	String	Error source, must be *reverse_side*
type	String	The section of the user's Telegram Passport which has the issue, one of “driver_license”, “identity_card”
file_hash	String	Base64-encoded hash of the file with the reverse side of the document
message	String	Error message

PassportElementErrorSelfie

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

Field	Type	Description
source	String	Error source, must be *selfie*
type	String	The section of the user's Telegram Passport which has the issue, one of “passport”, “driver_license”, “identity_card”, “internal_passport”
file_hash	String	Base64-encoded hash of the file with the selfie
message	String	Error message

PassportElementErrorFile

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

Field	Type	Description
source	String	Error source, must be *file*
type	String	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	String	Base64-encoded file hash
message	String	Error message

PassportElementErrorFiles

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

Field	Type	Description
source	String	Error source, must be *files*
type	String	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	Array of String	List of base64-encoded file hashes
message	String	Error message

PassportElementErrorTranslationFile

Represents an issue with one of the files that constitute the translation of a document. The error is considered resolved when the file changes.

Field	Type	Description
source	String	Error source, must be *translation_file*
type	String	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	String	Base64-encoded file hash
message	String	Error message

PassportElementErrorTranslationFiles

Represents an issue with the translated version of a document. The error is considered resolved when a file with the document translation change.

Field	Type	Description
source	String	Error source, must be *translation_files*
type	String	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	Array of String	List of base64-encoded file hashes
message	String	Error message

PassportElementErrorUnspecified

Represents an issue in an unspecified place. The error is considered resolved when new data is added.

Field	Type	Description
source	String	Error source, must be *unspecified*
type	String	Type of element of the user's Telegram Passport which has the issue
element_hash	String	Base64-encoded element hash
message	String	Error message

Games

Your bot can offer users **HTML5 games** to play solo or to compete against each other in groups and one-on-one chats. Create games via @BotFather using the */newgame* command. Please note that this kind of power requires responsibility: you will need to accept the terms for each game that your bots will be offering.

- Games are a new type of content on Telegram, represented by the Game and InlineQueryResultGame objects.
- Once you've created a game via BotFather, you can send games to chats as regular messages using the sendGame method, or use inline mode with InlineQueryResultGame.
- If you send the game message without any buttons, it will automatically have a 'Play *GameName*' button. When this button is pressed, your bot gets a CallbackQuery with the *game_short_name* of the requested game. You provide the correct URL for this particular user and the app opens the game in the in-app browser.
- You can manually add multiple buttons to your game message. Please note that the first button in the first row **must always** launch the game, using the field *callback_game* in InlineKeyboardButton. You can add extra buttons according to taste: e.g., for a description of the rules, or to open the game's official community.
- To make your game more attractive, you can upload a GIF animation that demostrates the game to the users via BotFather (see Lumberjack for example).
- A game message will also display high scores for the current chat. Use setGameScore to post high scores to the chat with the game, add the *edit_message* parameter to automatically update the message with the current scoreboard.
- Use getGameHighScores to get data for in-game high score tables.
- You can also add an extra sharing button for users to share their best score to different chats.
- For examples of what can be done using this new stuff, check the @gamebot and @gamee bots.

sendGame

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

Parameters	Type	Required	Description

chat_id	Integer	Yes	Unique identifier for the target chat
game_short_name	String	Yes	Short name of the game, serves as the unique identifier for the game. Set up your games via Botfather.
disable_notification	Boolean	Optional	Sends the message silently. Users will receive a notification with no sound.
reply_to_message_id	Integer	Optional	If the message is a reply, ID of the original message
reply_markup	InlineKeyboardMarkup	Optional	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.

Game

This object represents a game. Use BotFather to create and edit games, their short names will act as unique identifiers.

Field	Type	Description
title	String	Title of the game
description	String	Description of the game
photo	Array of PhotoSize	Photo that will be displayed in the game message in chats.
text	String	*Optional*. Brief description of the game or high scores included in the game message. Can be automatically edited to include current high scores for the game when the bot calls setGameScore, or manually edited using editMessageText. 0-4096 characters.
text_entities	Array of MessageEntity	*Optional*. Special entities that appear in *text*, such as usernames, URLs, bot commands, etc.
animation	Animation	*Optional*. Animation that will be displayed in the game message in chats. Upload via BotFather

CallbackGame

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

setGameScore

Use this method to set the score of the specified user in a game. On success, if the message was sent by the bot, returns the edited Message, otherwise returns *True*. Returns an error, if the new score is not greater than the user's current score in the chat and *force* is *False*.

Parameters	Type	Required	Description
user_id	Integer	Yes	User identifier
score	Integer	Yes	New score, must be non-negative
force	Boolean	Optional	Pass True, if the high score is allowed to decrease. This can be useful when fixing mistakes or banning cheaters
disable_edit_message	Boolean	Optional	Pass True, if the game message should not be automatically edited to include the current scoreboard
chat_id	Integer	Optional	Required if *inline_message_id* is not specified. Unique identifier for the target chat

message_id	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message

getGameHighScores

Use this method to get data for high score tables. Will return the score of the specified user and several of his neighbors in a game. On success, returns an *Array* of GameHighScore objects.

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

Parameters	Type	Required	Description
user_id	Integer	Yes	Target user id
chat_id	Integer	Optional	Required if *inline_message_id* is not specified. Unique identifier for the target chat
message_id	Integer	Optional	Required if *inline_message_id* is not specified. Identifier of the sent message
inline_message_id	String	Optional	Required if *chat_id* and *message_id* are not specified. Identifier of the inline message

GameHighScore

This object represents one row of the high scores table for a game.

Field	Type	Description
position	Integer	Position in high score table for the game
user	User	User
score	Integer	Score

And that‘s about all we’ve got for now.
If you've got any questions, please check out our Bot FAQ »