

# Package: telegramR (via r-universe)

June 2, 2026

**Type** Package

**Title** Interact with the 'Telegram' 'MTPROTO' API

**Version** 0.0.1

**Description** Provides a full-featured client for the 'Telegram' 'MTPROTO' protocol (<<https://core.telegram.org/api>>), enabling programmatic access to 'Telegram' chats, channels, messages, media, and stories. Implements binary encoding and decoding of the 'Telegram' 'TL' (Type Language) schema, authentication (including two-factor), encrypted transport, and high-level helpers for downloading channel history and reactions at scale. Intended for social-science research and data collection tasks that require direct API access rather than the 'Bot API'.

**License** MIT + file LICENSE

**Encoding** UTF-8

**ByteCompile** no

**Depends** R (>= 3.5.0)

**Imports** Rcpp (>= 1.0.0), digest (>= 0.6.37), openssl (>= 2.2.2), base64enc (>= 0.1-3), jsonlite (>= 1.8.9), R6, future, httr, logger, promises, later, bitops, gmp, xml2, mime, tibble, dplyr, callr

**LinkingTo** Rcpp

**Suggests** data.table, testthat (>= 3.0.0), withr, exiftoolr, fs, getPass, magick, knitr, rmarkdown, ggplot2, lubridate, tidyr, covr, devtools, readr, stringr, stopwords, RColorBrewer, scales, stringi

**Config/testthat/edition** 3

**RoxygenNote** 7.3.3

**Collate** 'RcppExports.R' 'abstract.R' 'account.R' 'add\_user\_to\_chat.R' 'aes.R' 'aesctr.R' 'authenticator.R' 'authkey.R' 'binaryreader.R' 'channel\_downloads.R' 'tlobjct.R' 'types.R' 'chats.R' 'common.R' 'compat\_promises.R' 'connection.R' 'dialogs.R' 'downloads.R' 'entitycache.R' 'factorization.R'

'functions.R' 'functions\_account.R' 'functions\_auth.R'  
 'functions\_bots.R' 'functions\_channels.R'  
 'functions\_chatlists.R' 'functions\_contacts.R'  
 'functions\_folders.R' 'functions\_fragment.R' 'functions\_help.R'  
 'functions\_langpack.R' 'functions\_messages.R'  
 'functions\_payments.R' 'functions\_phone.R' 'functions\_photos.R'  
 'functions\_premium.R' 'functions\_smsjobs.R' 'functions\_stats.R'  
 'functions\_stickers.R' 'functions\_stories.R'  
 'functions\_updates.R' 'functions\_upload.R' 'functions\_users.R'  
 'gzipped.R' 'helpers.R' 'html.R' 'http.R' 'libssl.R'  
 'join\_channel.R' 'markdown.R' 'messagecontainer.R'  
 'messagepacker.R' 'messageparse.R' 'messages.R'  
 'mtprotoplainsender.R' 'mtprotosender.R' 'mtprotostate.R'  
 'namespace.R' 'password.R' 'phone\_number\_checker.R'  
 'requestiter.R' 'requests.R' 'requeststate.R' 'rpcresult.R'  
 'rsa.R' 'send\_audio.R' 'send\_document.R' 'send\_file.R'  
 'send\_message.R' 'send\_photo.R' 'send\_video.R' 'tcpabridget.R'  
 'tcpfull.R' 'tcpintermediate.R' 'tcpmtproxy.R'  
 'tcpobfuscated.R' 'telegrambaseclient.R' 'telegramclient.R'  
 'tmessage.R' 'updates.R' 'users\_profile.R' 'utils.R'  
 'check\_invite\_link.R' 'increase\_view\_count.R' 'utils\_bridge.R'  
 'zzz.R'

**URL** <https://romankyrychenko.github.io/telegramR/>,  
<https://github.com/RomanKyrychenko/telegramR>

**BugReports** <https://github.com/RomanKyrychenko/telegramR/issues>

**VignetteBuilder** knitr

**NeedsCompilation** yes

**Author** Roman Kyrychenko [aut, cre, cph]

**Maintainer** Roman Kyrychenko <roman.kyrychenko@helsinki.fi>

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-06-02 08:00:02 UTC

**RemoteUrl** <https://github.com/cran/telegramR>

**RemoteRef** HEAD

**RemoteSha** e0cf02bd2690194de524b274ea914c5713b352ee

## Contents

+raw	3
+raw_bytes	4
add_user_to_chat	4
batch_download_channels	5
check_invite_link	6
check_phone_on_telegram	7

- check\_phones\_on\_telegram . . . . . 8
- check\_username\_on\_telegram . . . . . 8
- check\_usernames\_on\_telegram . . . . . 9
- download\_channel\_info . . . . . 10
- download\_channel\_media . . . . . 11
- download\_channel\_members . . . . . 12
- download\_channel\_messages . . . . . 13
- download\_channel\_reactions . . . . . 14
- download\_channel\_replies . . . . . 15
- estimate\_channel\_post\_count . . . . . 16
- get\_full\_user . . . . . 17
- increase\_view\_count . . . . . 17
- join\_channel . . . . . 18
- join\_private\_chat . . . . . 18
- join\_public\_channel . . . . . 19
- phone\_number\_checker . . . . . 19
- send\_audio . . . . . 20
- send\_document . . . . . 20
- send\_file . . . . . 21
- send\_message . . . . . 21
- send\_photo . . . . . 22
- send\_video . . . . . 22
- TelegramBaseClient . . . . . 23
- TelegramClient . . . . . 23
- update\_profile . . . . . 24
- update\_profile\_photo . . . . . 25
- update\_username . . . . . 25

**Index** 26

---

<code>+.raw</code>	<i>Addition operator for raw (fallback)</i>
--------------------	---

---

**Description**

This is tricky because we can't easily override '+' for base 'raw' type. But if the first operand is 'raw\_bytes', it works. If the first operand is 'raw', we might need to cast it.

**Usage**

```
## S3 method for class 'raw'
e1 + e2
```

**Arguments**

- e1           Left operand (raw vector).
- e2           Right operand (raw vector).

**Value**

A new raw\_bytes object.

---

<code>+.raw_bytes</code>	<i>Addition operator for raw_bytes</i>
--------------------------	--

---

**Description**

Allows using '+' to concatenate raw\_bytes objects, mimicking Python's byte concatenation.

**Usage**

```
## S3 method for class 'raw_bytes'
e1 + e2
```

**Arguments**

<code>e1</code>	Left operand.
<code>e2</code>	Right operand.

**Value**

A new raw\_bytes object.

---

<code>add_user_to_chat</code>	<i>Add a user to a chat or channel</i>
-------------------------------	--

---

**Description**

Add a user to a chat or channel

**Usage**

```
add_user_to_chat(client, chat, user, fwd_limit = 10)
```

**Arguments**

<code>client</code>	TelegramClient instance.
<code>chat</code>	Chat/channel (username, link, or entity).
<code>user</code>	User (username, link, or entity).
<code>fwd_limit</code>	Forward limit for basic chats. Default is 10.

**Value**

The API response.

---

 batch\_download\_channels

*Batch Download Multiple Telegram Channels*


---

## Description

Downloads info and messages for multiple channels in sequence, running each channel in an isolated callr subprocess so crashes cannot corrupt the parent session. Supports resume: channels already present in msgs\_file are skipped when skip\_completed = TRUE.

## Usage

```
batch_download_channels(
  channels,
  session,
  api_id,
  api_hash,
  info_file = NULL,
  msgs_file = NULL,
  reactions_file = NULL,
  replies_file = NULL,
  start_date = NULL,
  limit = Inf,
  timeout_sec = 30,
  max_timeouts = 5,
  chunk_size = 5000L,
  skip_completed = TRUE,
  dedup = TRUE,
  pkg_path = NULL,
  workers = 1L,
  verbose = TRUE
)
```

## Arguments

channels	character vector. Channel usernames (with or without "@") or numeric ids.
session	character. Path to the Telegram session file (passed to TelegramClient\$new()).
api_id	integer. Telegram API id.
api_hash	character. Telegram API hash.
info_file	character. Required. Path to the CSV file for channel info rows (appended to).
msgs_file	character. Required. Path to the CSV file for message rows (appended to, streamed in chunk_size chunks to avoid RAM accumulation).
reactions_file	character or NULL. If set, reaction counts are written to this CSV file.
replies_file	character or NULL. If set, reply counts are written to this CSV file.

start_date	character/Date/POSIXct or NULL. Passed to download_channel_messages().
limit	integer or Inf. Max messages per channel.
timeout_sec	numeric. Per-fetch timeout in seconds.
max_timeouts	integer. Timeouts before aborting a single channel.
chunk_size	integer. Rows buffered before each CSV flush.
skip_completed	logical. If TRUE (default), skip channels whose username already appears in msgs_file.
dedup	logical. If TRUE (default), skip messages already present in msgs_file based on message_id.
pkg_path	character or NULL. Path to the package root; passed to devtools::load_all() inside the subprocess. When NULL (default), library(telegramR) is used instead.
workers	integer. Number of parallel workers. Default 1L (sequential).
verbose	logical. If TRUE (default), print progress messages.

**Value**

A tibble with columns channel, status ("ok"/"skipped"/"error"), rows\_downloaded, error\_message.

---

check_invite_link	<i>Check a chat invite link without joining</i>
-------------------	---

---

**Description**

Check a chat invite link without joining

**Usage**

```
check_invite_link(client, invite_link)
```

**Arguments**

client	TelegramClient instance.
invite_link	Invite link or invite hash.

**Value**

The API response.

---

`check_phone_on_telegram`*Check whether a single phone number is registered on Telegram*

---

### Description

Adds the phone number to the authenticated account's contacts via 'ImportContactsRequest', captures any matching user, then removes the contact via 'DeleteContactsRequest'. The contact addition is transient — Telegram users are not notified.

### Usage

```
check_phone_on_telegram(  
    client,  
    phone,  
    download_profile_photo = FALSE,  
    photo_dir = NULL  
)
```

### Arguments

<code>client</code>	An authenticated [TelegramClient] instance.
<code>phone</code>	Phone number string in international format (e.g. "+15551234567").
<code>download_profile_photo</code>	If 'TRUE', attempt to download the matched user's profile photo into 'photo_dir' (default: 'tempdir()'). Failures are reported as warnings.
<code>photo_dir</code>	Directory to save downloaded profile photos; created if it doesn't exist. Ignored when 'download_profile_photo' is 'FALSE'.

### Value

A named list with the user's public fields, or a list with an 'error' field describing why the lookup failed (no match, multiple matches, RPC error, ...).

### See Also

[check\_phones\_on\_telegram()] for batch lookups.

---

 check\_phones\_on\_telegram

*Check several phone numbers in one call*


---

### Description

Convenience wrapper around [check\_phone\_on\_telegram()] that accepts either a character vector or a single comma-separated string (matching the upstream Python CLI). Duplicates are silently de-duplicated.

### Usage

```
check_phones_on_telegram(
  client,
  phones,
  download_profile_photo = FALSE,
  photo_dir = NULL
)
```

### Arguments

client	An authenticated [TelegramClient] instance.
phones	Character vector of phone numbers, or a single comma-separated string.
download_profile_photo, photo_dir	See [check_phone_on_telegram()].

### Value

A tibble with one row per unique phone number. Successful lookups expose the user's public fields; failures expose an 'error' column.

### See Also

[check\_phone\_on\_telegram()] for the single-phone variant.

---

 check\_username\_on\_telegram

*Check whether a single username belongs to a Telegram user*


---

### Description

Looks up the username via 'client\$get\_entity()'. Channels and group chats produce an 'error' row — this helper is for user accounts only.

**Usage**

```
check_username_on_telegram(
    client,
    username,
    download_profile_photo = FALSE,
    photo_dir = NULL
)
```

**Arguments**

`client` An authenticated [TelegramClient] instance.

`username` Username, with or without a leading '@'.

`download_profile_photo, photo_dir` See [check\_phone\_on\_telegram()].

**Value**

A named list with the user's public fields, or a list with an 'error' field.

**See Also**

[check\_usernames\_on\_telegram()] for batch lookups.

---

check\_usernames\_on\_telegram

*Check several usernames in one call*

---

**Description**

Convenience wrapper around [check\_username\_on\_telegram()].

**Usage**

```
check_usernames_on_telegram(
    client,
    usernames,
    download_profile_photo = FALSE,
    photo_dir = NULL
)
```

**Arguments**

`client` An authenticated [TelegramClient] instance.

`usernames` Character vector of usernames, or a single comma-separated string. Leading '@' is optional and stripped.

`download_profile_photo, photo_dir` See [check\_phone\_on\_telegram()].

**Value**

A tibble with one row per unique username.

**See Also**

[check\_username\_on\_telegram()] for the single-username variant.

---

download\_channel\_info *Download Full Channel Info By Channel*

---

**Description**

Fetches channel entity and full channel info by username or numeric id and returns a tibble.

**Usage**

```
download_channel_info(
  client,
  channel,
  region = NULL,
  include_raw = FALSE,
  timeout_sec = getOption("telegramR.iter_timeout", 60),
  reconnect_on_timeout = TRUE,
  max_attempts = 3,
  skip_estimate = FALSE,
  hard_timeout_sec = NULL
)
```

**Arguments**

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
region	character or NULL. Optional region tag to attach.
include_raw	logical. If TRUE, include raw Telegram objects as list columns.
timeout_sec	numeric. Timeout in seconds for network operations.
reconnect_on_timeout	logical. Reconnect on timeout if TRUE.
max_attempts	integer. Number of retry attempts for resolving and full info.
skip_estimate	logical. If TRUE, skip estimate step for last_message_id.
hard_timeout_sec	numeric or NULL. Optional hard time limit for the whole call.

**Value**

A tibble with flattened channel info and optional list columns for raw objects.

---

 download\_channel\_media

*Download Channel Media By Channel*


---

## Description

Downloads media (photos/videos/documents) from a channel by username or numeric id and returns a tibble with file paths.

## Usage

```
download_channel_media(
  client,
  channel,
  limit = Inf,
  start_date = NULL,
  end_date = NULL,
  media_types = c("photo", "video", "image", "document"),
  out_dir = NULL,
  show_progress = TRUE,
  wait_time = 0,
  retries = 1,
  include_errors = TRUE,
  use_original_filename = FALSE,
  ...
)
```

## Arguments

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
limit	integer or Inf. Maximum number of messages to fetch.
start_date	POSIXct/Date/character. Earliest date to include (UTC).
end_date	POSIXct/Date/character. Latest date to include (UTC).
media_types	character vector. Media types to download (e.g. "photo", "video", "document", "image", "audio").
out_dir	character. Required. Directory to save files into (e.g. tempdir()).
show_progress	logical. If TRUE, display a progress bar.
wait_time	numeric. Seconds to sleep between requests to avoid flood waits.
retries	integer. Number of retries per media download on failure.
include_errors	logical. If TRUE, include error messages per row.
use_original_filename	logical. If TRUE and a document has a filename attribute, use it for the saved file name.
...	Passed to client\$iter_messages() (e.g. offset_id, max_id, min_id).

**Value**

A tibble with message\_id, channel info, media\_type, file\_path, and original\_filename.

---

download\_channel\_members

*Download Channel Members By Channel*

---

**Description**

Fetches channel members (participants) by username or numeric id.

**Usage**

```
download_channel_members(
  client,
  channel,
  limit = Inf,
  search = "",
  filter = NULL,
  include_channel = TRUE,
  show_progress = TRUE
)
```

**Arguments**

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
limit	integer or Inf. Maximum number of members to fetch.
search	character. Search query for participant names/usernames.
filter	A participants filter, e.g. ChannelParticipantsAdmins.
include_channel	logical. If TRUE, include channel fields on every row.
show_progress	logical. If TRUE, display a progress bar.

**Value**

A tibble.

---

 download\_channel\_messages

*Download Channel Messages By Channel*


---

## Description

Fetches messages for a channel by username or numeric id and returns a tibble with message fields and nested structures as list columns.

## Usage

```
download_channel_messages(
  client,
  channel,
  limit = Inf,
  include_channel = TRUE,
  start_date = NULL,
  end_date = NULL,
  since_message_id = NULL,
  show_progress = TRUE,
  timeout_sec = getOption("telegramR.iter_timeout", 60),
  max_timeouts = 3,
  reconnect_on_timeout = TRUE,
  output_file = NULL,
  chunk_size = 5000L,
  partial_on_error = TRUE,
  ...
)
```

## Arguments

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
limit	integer or Inf. Maximum number of messages to fetch.
include_channel	logical. If TRUE, include channel fields on every row.
start_date	POSIXct/Date/character. Earliest date to include (UTC).
end_date	POSIXct/Date/character. Latest date to include (UTC).
since_message_id	integer or NULL. If set, only fetch messages with id > this value (incremental/resume download). Maps to min_id in the API.
show_progress	logical. If TRUE, display a progress bar.
timeout_sec	numeric. Max seconds to wait per fetch before retrying. Use 0 to disable.
max_timeouts	integer. Number of timeouts to tolerate before aborting.

reconnect_on_timeout	logical. If TRUE, reconnect client on timeout.
output_file	character or NULL. When set, rows are written to this CSV file in chunks rather than accumulated in memory. Returns the row count invisibly instead of a tibble. Appends to the file if it already exists (adds header only on first write).
chunk_size	integer. Number of rows to buffer before each write when output_file is set. Default 5000.
partial_on_error	logical. If TRUE (default), non-timeout errors emit a warning and return the rows collected so far rather than throwing. If FALSE, errors propagate immediately.
...	Passed to client\$iter_messages() (e.g. offset_id, max_id, min_id).

**Value**

A tibble (or invisible row count when output\_file is set).

---

download\_channel\_reactions

*Download Channel Reactions By Channel*

---

**Description**

Fetches message reactions for a channel by username or numeric id.

**Usage**

```
download_channel_reactions(
  client,
  channel,
  limit = Inf,
  start_date = NULL,
  end_date = NULL,
  include_channel = TRUE,
  show_progress = TRUE,
  output_file = NULL,
  chunk_size = 5000L,
  ...
)
```

**Arguments**

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
limit	integer or Inf. Maximum number of messages to fetch.
start_date	POSIXct/Date/character. Earliest date to include (UTC).

end_date	POSIXct/Date/character. Latest date to include (UTC).
include_channel	logical. If TRUE, include channel fields on every row.
show_progress	logical. If TRUE, display a progress bar.
output_file	character or NULL. When set, rows are written to this CSV file in chunks instead of being accumulated in memory.
chunk_size	integer. Number of rows to buffer before each write when output_file is set. Default 5000.
...	Passed to client\$iter_messages() (e.g. offset_id, max_id, min_id).

**Value**

A tibble.

---

download\_channel\_replies

*Download Channel Replies/Comments By Channel*

---

**Description**

Fetches replies (comments) to channel posts by username or numeric id.

**Usage**

```
download_channel_replies(
  client,
  channel,
  message_ids = NULL,
  message_limit = 100,
  reply_limit = Inf,
  include_channel = TRUE,
  show_progress = TRUE,
  output_file = NULL,
  chunk_size = 5000L,
  ...
)
```

**Arguments**

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
message_ids	integer vector or NULL. If NULL, replies are fetched for the most recent posts.
message_limit	integer. Number of recent posts to inspect when message_ids is NULL.
reply_limit	integer or Inf. Maximum number of replies per post.

include_channel	logical. If TRUE, include channel fields on every row.
show_progress	logical. If TRUE, display a progress bar.
output_file	character or NULL. When set, rows are written to this CSV file in chunks instead of being accumulated in memory.
chunk_size	integer. Number of rows to buffer before each write when output_file is set. Default 5000.
...	Passed to client\$iter_messages() when fetching recent posts.

**Value**

A tibble.

---

estimate\_channel\_post\_count  
*Estimate Channel Post Count (Approx)*

---

**Description**

Gets the latest message ID and uses it as an upper-bound estimate of total posts. This is approximate due to deletions and gaps in message IDs.

**Usage**

```
estimate_channel_post_count(
  client,
  channel,
  timeout_sec = getOption("telegramR.iter_timeout", 60),
  reconnect_on_timeout = TRUE
)
```

**Arguments**

client	TelegramClient instance.
channel	character or numeric. Channel username (with or without "@") or numeric id.
timeout_sec	numeric. Timeout in seconds for network operations.
reconnect_on_timeout	logical. Reconnect on timeout if TRUE.

**Value**

A tibble with last\_message\_id and a note.

---

get_full_user	<i>Retrieve full information about a user</i>
---------------	---

---

**Description**

Retrieve full information about a user

**Usage**

```
get_full_user(client, user = "me")
```

**Arguments**

client	TelegramClient instance.
user	User (username, id, link) or "me".

**Value**

The API response (UserFull).

---

increase_view_count	<i>Increase view count for channel messages</i>
---------------------	---

---

**Description**

Increase view count for channel messages

**Usage**

```
increase_view_count(client, channel, message_ids)
```

**Arguments**

client	TelegramClient instance.
channel	Channel (username, link, or entity).
message_ids	Numeric vector of message IDs.

**Value**

The API response (views).

---

join_channel	<i>Join a chat or channel by username or invite link</i>
--------------	--

---

**Description**

Join a chat or channel by username or invite link

**Usage**

```
join_channel(client, channel)
```

**Arguments**

client	TelegramClient instance.
channel	Username/link or invite link.

**Value**

The API response.

---

join_private_chat	<i>Join a private chat or channel by invite link</i>
-------------------	--

---

**Description**

Join a private chat or channel by invite link

**Usage**

```
join_private_chat(client, invite_link)
```

**Arguments**

client	TelegramClient instance.
invite_link	Invite link (e.g. <a href="https://t.me/joinchat/...">https://t.me/joinchat/...</a> or <a href="https://t.me/+...">https://t.me/+...</a> ).

**Value**

The API response.

---

join\_public\_channel    *Join a public channel by username*

---

### Description

Join a public channel by username

### Usage

```
join_public_channel(client, channel)
```

### Arguments

client	TelegramClient instance.
channel	Public channel username or link (non-invite).

### Value

The API response.

---

phone\_number\_checker    *Check whether phone numbers or usernames belong to Telegram accounts*

---

### Description

These helpers port the functionality of bellingcat's *telegram-phone-number-checker* Python tool to R. They use the currently authenticated [TelegramClient] to look up users by phone number (via 'ImportContactsRequest' / 'DeleteContactsRequest') or by username (via 'client\$get\_entity()'), and return the public profile fields Telegram returns about the matched user.

### Details

Phone-number lookups briefly add the number to your contact list and then delete it. Telegram throttles abusive callers — Bellingcat recommends using a fresh, residential-IP account rather than a personal one.

---

send_audio	<i>Send an Audio File</i>
------------	---------------------------

---

**Description**

Convenience wrapper to send an audio file to a target entity.

**Usage**

```
send_audio(client, entity, file, ...)
```

**Arguments**

client	TelegramClient instance.
entity	Target entity (username, id, or TLObject).
file	Audio file path or file-like object.
...	Additional arguments passed to 'TelegramClient\$send_message()' (e.g., 'message', 'attributes', 'thumb', 'parse_mode', 'buttons').

**Value**

The sent message object (or Updates result).

---

send_document	<i>Send a Document</i>
---------------	------------------------

---

**Description**

Convenience wrapper to send a document (file) to a target entity.

**Usage**

```
send_document(client, entity, file, ...)
```

**Arguments**

client	TelegramClient instance.
entity	Target entity (username, id, or TLObject).
file	File path or file-like object.
...	Additional arguments passed to 'TelegramClient\$send_message()' (e.g., 'message', 'attributes', 'thumb', 'parse_mode', 'buttons').

**Value**

The sent message object (or Updates result).

---

send_file	<i>Send a File</i>
-----------	--------------------

---

**Description**

Convenience wrapper around ‘TelegramClient\$send\_message()’ that sends a file and resolves futures/promises.

**Usage**

```
send_file(client, entity, file, ...)
```

**Arguments**

client	TelegramClient instance.
entity	Target entity (username, id, or TLObject).
file	File path or file-like object.
...	Additional arguments passed to ‘TelegramClient\$send_message()’ (e.g., ‘message’, ‘force_document’, ‘attributes’, ‘thumb’, ‘parse_mode’).

**Value**

The sent message object (or Updates result).

---

send_message	<i>Send a Message</i>
--------------	-----------------------

---

**Description**

Convenience wrapper around ‘TelegramClient\$send\_message()’ that resolves futures/promises and returns the sent message (or Updates object).

**Usage**

```
send_message(client, entity, ...)
```

**Arguments**

client	TelegramClient instance.
entity	Target entity (username, id, or TLObject).
...	Passed to ‘TelegramClient\$send_message()’.

**Value**

The sent message object (or Updates result).

---

send_photo	<i>Send a Photo</i>
------------	---------------------

---

**Description**

Convenience wrapper to send a photo to a target entity.

**Usage**

```
send_photo(client, entity, file, ...)
```

**Arguments**

client	TelegramClient instance.
entity	Target entity (username, id, or TLObject).
file	Photo file path or file-like object.
...	Additional arguments passed to 'TelegramClient\$send_message()' (e.g., 'message', 'parse_mode', 'buttons').

**Value**

The sent message object (or Updates result).

---

send_video	<i>Send a Video</i>
------------	---------------------

---

**Description**

Convenience wrapper to send a video to a target entity.

**Usage**

```
send_video(client, entity, file, ...)
```

**Arguments**

client	TelegramClient instance.
entity	Target entity (username, id, or TLObject).
file	Video file path or file-like object.
...	Additional arguments passed to 'TelegramClient\$send_message()' (e.g., 'message', 'supports_streaming', 'parse_mode', 'buttons').

**Value**

The sent message object (or Updates result).

---

TelegramBaseClient      *TelegramBaseClient*

---

**Description**

Low-level Telegram MTProto client. Most users should use TelegramClient.

**Usage**

```
TelegramBaseClient
```

**Details**

This is an R6 class providing lower-level transport and protocol operations.

**Value**

An R6 object of class TelegramBaseClient.

**See Also**

[TelegramClient](#)

**Examples**

```
## Not run:
client <- TelegramClient$new("my_session", api_id = 123, api_hash = "abc")
client$connect()

## End(Not run)
```

---

TelegramClient      *TelegramClient*

---

**Description**

High-level Telegram MTProto client. Use TelegramClient\$new() to create a session and connect.

**Usage**

```
TelegramClient
```

**Details**

This is an R6 class. Typical usage:

```
client <- TelegramClient$new("my_session", api_id = 123, api_hash = "...")
client$connect()
```

**Value**

An R6 object of class TelegramClient.

**See Also**

[TelegramBaseClient](#)

**Examples**

```
## Not run:  
client <- TelegramClient$new("my_session", api_id = 123, api_hash = "...")  
client$connect()  
  
## End(Not run)
```

---

update_profile	<i>Update your name and/or bio</i>
----------------	------------------------------------

---

**Description**

Update your name and/or bio

**Usage**

```
update_profile(client, first_name = NULL, last_name = NULL, bio = NULL)
```

**Arguments**

client	TelegramClient instance.
first_name	First name (optional).
last_name	Last name (optional).
bio	About/bio text (optional).

**Value**

The API response.

---

update\_profile\_photo    *Update your profile photo*

---

**Description**

Update your profile photo

**Usage**

```
update_profile_photo(client, file, progress_callback = NULL)
```

**Arguments**

client	TelegramClient instance.
file	Path to image file.
progress_callback	Optional callback (pos, total).

**Value**

The API response.

---

update\_username        *Update your username*

---

**Description**

Update your username

**Usage**

```
update_username(client, username)
```

**Arguments**

client	TelegramClient instance.
username	New username (without @).

**Value**

The API response.

# Index

`+.raw`, [3](#)  
`+.raw_bytes`, [4](#)  
`add_user_to_chat`, [4](#)  
`batch_download_channels`, [5](#)  
`check_invite_link`, [6](#)  
`check_phone_on_telegram`, [7](#)  
`check_phones_on_telegram`, [8](#)  
`check_username_on_telegram`, [8](#)  
`check_usernames_on_telegram`, [9](#)  
`download_channel_info`, [10](#)  
`download_channel_media`, [11](#)  
`download_channel_members`, [12](#)  
`download_channel_messages`, [13](#)  
`download_channel_reactions`, [14](#)  
`download_channel_replies`, [15](#)  
`estimate_channel_post_count`, [16](#)  
`get_full_user`, [17](#)  
`increase_view_count`, [17](#)  
`join_channel`, [18](#)  
`join_private_chat`, [18](#)  
`join_public_channel`, [19](#)  
`phone_number_checker`, [19](#)  
`send_audio`, [20](#)  
`send_document`, [20](#)  
`send_file`, [21](#)  
`send_message`, [21](#)  
`send_photo`, [22](#)  
`send_video`, [22](#)  
`TelegramBaseClient`, [23](#), [24](#)  
`TelegramClient`, [23](#), [23](#)  
`update_profile`, [24](#)  
`update_profile_photo`, [25](#)  
`update_username`, [25](#)