Package: telegramR (via r-universe)

October 9, 2026

Type Package

Title Interact with the 'Telegram' 'MTProto' API

Version 0.0.2

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

SystemRequirements OpenSSL (>= 1.0.1): libssl-dev (deb), openssl-devel (rpm), openssl@3 (brew)

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, promises, later, bitops, gmp, xml2, mime, tibble, dplyr, callr, cli (>= 3.4.0), compiler

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

Roxygen list(roclets = c(``collate'', ``namespace'', ``rd''))

Collate 'RcppExports.R' 'abstract.R' 'account.R' 'add_user_to_chat.R'
'aes.R' 'aesctr.R' 'authentificator.R' 'authkey.R'
'binaryreader.R' 'channel_downloads.R' 'tlobject.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'
'gzippacked.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'
'tlmessage.R' 'tl_fast.R' 'updates.R' 'users_profile.R'
'utils.R' 'check_invite_link.R' 'increase_view_count.R'
'utils_bridge.R' 'ui.R' 'zzz.R'

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

BugReports https://github.com/RomanKyrychenko/telegramR/issues

VignetteBuilder knitr

Config/pak/sysreqs libgmp3-dev libxml2-dev libssl-dev

Repository https://romankyrychenko.r-universe.dev

Date/Publication 2026-10-09 06:58:50 UTC

RemoteUrl https://github.com/romankyrychenko/telegramr

RemoteRef HEAD

RemoteSha eb0d0f5ec6724cb58eaab4199c451fbd440ad7d9

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
TelegramClient 23
update_profile 44
update_profile_photo 44
update_username 45

Index **46**

`+.raw` *Addition operator for raw (fallback)*

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.

+.raw_bytes *Addition operator for raw_bytes*

Description

Allows using '+' to concatenate raw_bytes objects, mimicking Python's byte concatenation.

Usage

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

Arguments

e1	Left operand.
e2	Right operand.

Value

A new raw_bytes object.

---

add_user_to_chat *Add a user to a chat or channel*

Description

Add a user to a chat or channel

Usage

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

Arguments

client	TelegramClient instance.
chat	Chat/channel (username, link, or entity).
user	User (username, link, or entity).
fwd_limit	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 *Check a chat invite link without joining*

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

client	An authenticated [TelegramClient] instance.
phone	Phone number string in international format (e.g. '"+15551234567"').
download_profile_photo	If 'TRUE', attempt to download the matched user's profile photo into 'photo_dir' (default: 'tempdir()'). Failures are reported as warnings.
photo_dir	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 *Retrieve full information about a user*

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 *Increase view count for channel messages*

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 *Join a chat or channel by username or invite link*

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 *Join a private chat or channel by invite link*

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. https://t.me/joinchat/... or https://t.me/+...).

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 *Send an Audio File*

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 *Send a Document*

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 *Send a File*

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 *Send a Message*

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 *Send a Photo*

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 *Send a Video*

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).

TelegramClient *TelegramClient*

Description

High-level 'Telegram' 'MTProto' client. Create one with `TelegramClient$new(session, api_id, api_hash)`, then `$start()` (interactive login) or `$connect()` together with `$send_code_request()` and `$sign_in()` to authenticate. Once connected it exposes the messaging, channel, media and download methods used throughout the package documentation.

Details

This is an R6 class. Typical usage:

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

Value

An R6 generator object of class `TelegramClient`.

Super class

`telegramR::TelegramBaseClient` -> `TelegramClient`

Methods

Public methods:

- `TelegramClient$takeout()`
- `TelegramClient$end_takeout()`
- `TelegramClient$start()`
- `TelegramClient$sign_in()`
- `TelegramClient$sign_up()`
- `TelegramClient$send_code_request()`
- `TelegramClient$qr_login()`
- `TelegramClient$log_out()`
- `TelegramClient$edit_2fa()`
- `TelegramClient$parse_phone_and_hash()`
- `TelegramClient$on_login()`
- `TelegramClient$start_impl()`
- `TelegramClient$parse_phone()`
- `TelegramClient$compute_check()`
- `TelegramClient$compute_digest()`
- `TelegramClient$get_display_name()`
- `TelegramClient$download_profile_photo()`

- TelegramClient$download_media()
- TelegramClient$download_file()
- TelegramClient$.download_file()
- TelegramClient$iter_download()
- TelegramClient$.iter_download()
- TelegramClient$get_thumb()
- TelegramClient$download_cached_photo_size()
- TelegramClient$download_photo()
- TelegramClient$get_kind_and_names()
- TelegramClient$download_document()
- TelegramClient$download_contact()
- TelegramClient$download_web_document()
- TelegramClient$get_proper_filename()
- TelegramClient$iter_dialogs()
- TelegramClient$get_dialogs()
- TelegramClient$iter_drafts()
- TelegramClient$get_drafts()
- TelegramClient$edit_folder()
- TelegramClient$delete_dialog()
- TelegramClient$conversation()
- TelegramClient$iter_participants()
- TelegramClient$get_participants()
- TelegramClient$iter_admin_log()
- TelegramClient$get_admin_log()
- TelegramClient$iter_profile_photos()
- TelegramClient$get_profile_photos()
- TelegramClient$action()
- TelegramClient$edit_admin()
- TelegramClient$edit_permissions()
- TelegramClient$kick_participant()
- TelegramClient$get_permissions()
- TelegramClient$get_stats()
- TelegramClient$inline_query()
- TelegramClient$invoke_function()
- TelegramClient$custom_inline_results()
- TelegramClient$iter_messages()
- TelegramClient$iter_messages_page()
- TelegramClient$get_messages()
- TelegramClient$channels_get_messages()
- TelegramClient$collect()
- TelegramClient$collect_one()
- TelegramClient$send_message()

- TelegramClient$forward_messages()
- TelegramClient$edit_message()
- TelegramClient$delete_messages()
- TelegramClient$send_read_acknowledge()
- TelegramClient$pin_message()
- TelegramClient$unpin_message()
- TelegramClient$get_comment_data()
- TelegramClient$pin_internal()
- TelegramClient$send_file()
- TelegramClient$upload_file()
- TelegramClient$file_to_media()
- TelegramClient$resize_photo_if_needed()
- TelegramClient$is_image()
- TelegramClient$get_appropriated_part_size()
- TelegramClient$invoke()
- TelegramClient$build_reply_markup()
- TelegramClient$run_until_disconnected()
- TelegramClient$set_receive_updates()
- TelegramClient$on()
- TelegramClient$add_event_handler()
- TelegramClient$remove_event_handler()
- TelegramClient$list_event_handlers()
- TelegramClient$catch_up()
- TelegramClient$update_loop()
- TelegramClient$preprocess_updates()
- TelegramClient$keepalive_loop()
- TelegramClient$dispatch_update()
- TelegramClient$handle_auto_reconnect()
- TelegramClient$set_parse_mode()
- TelegramClient$get_parse_mode()
- TelegramClient$sanitize_parse_mode()
- TelegramClient$parse_message_text()
- TelegramClient$replace_with_mention()
- TelegramClient$get_response_message()
- TelegramClient$call()
- TelegramClient$call_internal()
- TelegramClient$get_me()
- TelegramClient$self_id()
- TelegramClient$is_bot()
- TelegramClient$is_user_authorized()
- TelegramClient$get_entity()
- TelegramClient$get_input_entity()

- TelegramClient$getInputEntity()
- TelegramClient$getInputPeer()
- TelegramClient$getInputUser()
- TelegramClient$getInputChannel()
- TelegramClient$getInputMessage()
- TelegramClient$getInputMedia()
- TelegramClient$getInputDocument()
- TelegramClient$getInputPhoto()
- TelegramClient$getInputChatPhoto()
- TelegramClient$getInputGroupCall()
- TelegramClient$get_peer()
- TelegramClient$get_peer_id()
- TelegramClient$get_entity_from_string()
- TelegramClient$get_input_dialog()
- TelegramClient$get_input_notify()
- TelegramClient$download_files_parallel()
- TelegramClient$new()
- TelegramClient$clone()

Method `takeout()`**:**

Usage:

```
TelegramClient$takeout(
  finalize = TRUE,
  contacts = NULL,
  users = NULL,
  chats = NULL,
  megagroups = NULL,
  channels = NULL,
  files = NULL,
  max_file_size = NULL
)
```

Method `end_takeout()`**:**

Usage:

```
TelegramClient$end_takeout(success)
```

Method `start()`**:**

Usage:

```
TelegramClient$start(
  phone = NULL,
  password = NULL,
  bot_token = NULL,
  force_sms = FALSE,
  code_callback = NULL,
  first_name = "New User",
```

```
  last_name = "",
  max_attempts = 3
)
```

Method `sign_in()`**:**

Usage:

```
TelegramClient$sign_in(
  phone = NULL,
  code = NULL,
  password = NULL,
  bot_token = NULL,
  phone_code_hash = NULL
)
```

Method `sign_up()`**:**

Usage:

```
TelegramClient$sign_up(
  code,
  first_name,
  last_name = "",
  phone = NULL,
  phone_code_hash = NULL
)
```

Method `send_code_request()`**:**

Usage:

```
TelegramClient$send_code_request(phone, force_sms = FALSE, retry_count = 0)
```

Method `qr_login()`**:**

Usage:

```
TelegramClient$qr_login(ignored_ids = NULL)
```

Method `log_out()`**:**

Usage:

```
TelegramClient$log_out()
```

Method `edit_2fa()`**:**

Usage:

```
TelegramClient$edit_2fa(
  current_password = NULL,
  new_password = NULL,
  hint = "",
  email = NULL,
  email_code_callback = NULL
)
```

Method `parse_phone_and_hash()`**:**

Usage:

```
TelegramClient$parse_phone_and_hash(phone, phone_hash)
```

Method `on_login()`**:**

Usage:

```
TelegramClient$on_login(user)
```

Method `start_impl()`**:**

Usage:

```
TelegramClient$start_impl(
  phone,
  password,
  bot_token,
  force_sms,
  code_callback,
  first_name,
  last_name,
  max_attempts
)
```

Method `parse_phone()`**:**

Usage:

```
TelegramClient$parse_phone(phone)
```

Method `compute_check()`**:**

Usage:

```
TelegramClient$compute_check(request, password)
```

Method `compute_digest()`**:**

Usage:

```
TelegramClient$compute_digest(algo, password)
```

Method `get_display_name()`**:**

Usage:

```
TelegramClient$get_display_name(user)
```

Method `download_profile_photo()`**:**

Usage:

```
TelegramClient$download_profile_photo(entity, file = NULL, download_big = TRUE)
```

Method `download_media()`**:**

Usage:

```
TelegramClient$download_media(
  message,
  file = NULL,
  thumb = NULL,
  progress_callback = NULL
)
```

Method `download_file()`**:**

Usage:

```
TelegramClient$download_file(
  input_location,
  file = NULL,
  part_size_kb = NULL,
  file_size = NULL,
  progress_callback = NULL,
  dc_id = NULL,
  key = NULL,
  iv = NULL,
  msg_data = NULL,
  cdn_redirect = NULL
)
```

Method `.download_file()`**:**

Usage:

```
TelegramClient$.download_file(
  input_location,
  file = NULL,
  part_size_kb = NULL,
  file_size = NULL,
  progress_callback = NULL,
  dc_id = NULL,
  key = NULL,
  iv = NULL,
  msg_data = NULL,
  cdn_redirect = NULL
)
```

Method `iter_download()`**:**

Usage:

```
TelegramClient$iter_download(
  file,
  offset = 0,
  stride = NULL,
  limit = NULL,
  chunk_size = NULL,
  request_size = MAX_CHUNK_SIZE,
  file_size = NULL,
  dc_id = NULL,
  msg_data = NULL,
  cdn_redirect = NULL
)
```

Method `.iter_download()`**:**

Usage:

```
TelegramClient$.iter_download(
  file,
  offset = 0,
  stride = NULL,
  limit = NULL,
  chunk_size = NULL,
  request_size = MAX_CHUNK_SIZE,
  file_size = NULL,
  dc_id = NULL,
  msg_data = NULL,
  cdn_redirect = NULL
)
```

Method `get_thumb()`**:**

Usage:

```
TelegramClient$get_thumb(thumbs, thumb)
```

Method `download_cached_photo_size()`**:**

Usage:

```
TelegramClient$download_cached_photo_size(size, file)
```

Method `download_photo()`**:**

Usage:

```
TelegramClient$download_photo(photo, file, date, thumb, progress_callback)
```

Method `get_kind_and_names()`**:**

Usage:

```
TelegramClient$get_kind_and_names(attributes)
```

Method `download_document()`**:**

Usage:

```
TelegramClient$download_document(
  document,
  file,
  date,
  thumb,
  progress_callback,
  msg_data
)
```

Method `download_contact()`**:**

Usage:

```
TelegramClient$download_contact(mm_contact, file)
```

Method `download_web_document()`**:**

Usage:

```
TelegramClient$download_web_document(web, file, progress_callback)
```

Method `get_proper_filename()`**:**

Usage:

```
TelegramClient$get_proper_filename(
  file,
  kind,
  extension,
  date = NULL,
  possible_names = NULL
)
```

Method `iter_dialogs()`**:**

Usage:

```
TelegramClient$iter_dialogs(
  limit = NULL,
  offset_date = NULL,
  offset_id = 0,
  offset_peer = InputPeerEmpty$new(),
  ignore_pinned = FALSE,
  ignore_migrated = FALSE,
  folder = NULL,
  archived = NULL
)
```

Method `get_dialogs()`**:**

Usage:

```
TelegramClient$get_dialogs(
  limit = NULL,
  offset_date = NULL,
  offset_id = 0,
  offset_peer = InputPeerEmpty$new(),
  ignore_pinned = FALSE,
  ignore_migrated = FALSE,
  folder = NULL,
  archived = NULL
)
```

Method `iter_drafts()`**:**

Usage:

```
TelegramClient$iter_drafts(entity = NULL)
```

Method `get_drafts()`**:**

Usage:

```
TelegramClient$get_drafts(entity = NULL)
```

Method `edit_folder()`**:**

Usage:

```
TelegramClient$edit_folder(entity = NULL, folder = NULL, unpack = NULL)
```

Method `delete_dialog()`**:**

Usage:

```
TelegramClient$delete_dialog(entity, revoke = FALSE)
```

Method `conversation()`**:**

Usage:

```
TelegramClient$conversation(
  entity,
  timeout = 60,
  total_timeout = NULL,
  max_messages = 100,
  exclusive = TRUE,
  replies_are_responses = TRUE
)
```

Method `iter_participants()`**:**

Usage:

```
TelegramClient$iter_participants(
  entity,
  limit = NULL,
  search = "",
  filter = NULL,
  aggressive = FALSE
)
```

Method `get_participants()`**:**

Usage:

```
TelegramClient$get_participants(...)
```

Method `iter_admin_log()`**:**

Usage:

```
TelegramClient$iter_admin_log(
  entity,
  limit = NULL,
  max_id = 0,
  min_id = 0,
  search = NULL,
  admins = NULL,
  join = NULL,
  leave = NULL,
  invite = NULL,
  restrict = NULL,
  unrestrict = NULL,
  ban = NULL,
  unban = NULL,
  promote = NULL,
  demote = NULL,
```

```
  info = NULL,
  settings = NULL,
  pinned = NULL,
  edit = NULL,
  delete = NULL,
  group_call = NULL
)
```

Method `get_admin_log()`**:**

Usage:

```
TelegramClient$get_admin_log(...)
```

Method `iter_profile_photos()`**:**

Usage:

```
TelegramClient$iter_profile_photos(
  entity,
  limit = NULL,
  offset = 0,
  max_id = 0
)
```

Method `get_profile_photos()`**:**

Usage:

```
TelegramClient$get_profile_photos(...)
```

Method `action()`**:**

Usage:

```
TelegramClient$action(entity, action, delay = 4, auto_cancel = TRUE)
```

Method `edit_admin()`**:**

Usage:

```
TelegramClient$edit_admin(
  entity,
  user,
  change_info = NULL,
  post_messages = NULL,
  edit_messages = NULL,
  delete_messages = NULL,
  ban_users = NULL,
  invite_users = NULL,
  pin_messages = NULL,
  add_admins = NULL,
  manage_call = NULL,
  anonymous = NULL,
  is_admin = NULL,
  title = NULL
)
```

Method `edit_permissions()`**:**

Usage:

```
TelegramClient$edit_permissions(
  entity,
  user = NULL,
  until_date = NULL,
  view_messages = TRUE,
  send_messages = TRUE,
  send_media = TRUE,
  send_stickers = TRUE,
  send_gifs = TRUE,
  send_games = TRUE,
  send_inline = TRUE,
  embed_link_previews = TRUE,
  send_polls = TRUE,
  change_info = TRUE,
  invite_users = TRUE,
  pin_messages = TRUE
)
```

Method `kick_participant()`**:**

Usage:

```
TelegramClient$kick_participant(entity, user)
```

Method `get_permissions()`**:**

Usage:

```
TelegramClient$get_permissions(entity, user = NULL)
```

Method `get_stats()`**:**

Usage:

```
TelegramClient$get_stats(entity, message = NULL)
```

Method `inline_query()`**:**

Usage:

```
TelegramClient$inline_query(
  bot,
  query,
  entity = NULL,
  offset = NULL,
  geo_point = NULL
)
```

Method `invoke_function()`**:**

Usage:

```
TelegramClient$invoke_function(function_name, params)
```

Method `custom_inline_results()`**:**

Usage:

```
TelegramClient$custom_inline_results(result, peer)
```

Method `iter_messages()`**:**

Usage:

```
TelegramClient$iter_messages(
  entity,
  limit = NULL,
  offset_date = NULL,
  offset_id = 0L,
  max_id = 0L,
  min_id = 0L,
  add_offset = 0L,
  search = NULL,
  filter = NULL,
  from_user = NULL,
  wait_time = NULL,
  ids = NULL,
  reverse = FALSE,
  reply_to = NULL,
  scheduled = FALSE
)
```

Method `iter_messages_page()`**:**

Usage:

```
TelegramClient$iter_messages_page(
  entity,
  limit = NULL,
  offset_date = NULL,
  offset_id = 0L,
  max_id = 0L,
  min_id = 0L,
  add_offset = 0L,
  search = NULL,
  filter = NULL,
  from_user = NULL,
  wait_time = NULL,
  reverse = FALSE,
  reply_to = NULL,
  scheduled = FALSE
)
```

Method `get_messages()`**:**

Usage:

```
TelegramClient$get_messages(...)
```

Method `channels_get_messages()`**:**

Usage:

```
TelegramClient$channels_get_messages(channel, ids)
```

Method `collect()`**:**

Usage:

```
TelegramClient$collect(it)
```

Method `collect_one()`**:**

Usage:

```
TelegramClient$collect_one(it)
```

Method `send_message()`**:**

Usage:

```
TelegramClient$send_message(
  entity,
  message = "",
  reply_to = NULL,
  attributes = NULL,
  parse_mode = NULL,
  formatting_entities = NULL,
  link_preview = TRUE,
  file = NULL,
  thumb = NULL,
  force_document = FALSE,
  clear_draft = FALSE,
  buttons = NULL,
  silent = NULL,
  background = NULL,
  supports_streaming = FALSE,
  schedule = NULL,
  comment_to = NULL,
  nosound_video = NULL,
  send_as = NULL,
  message_effect_id = NULL
)
```

Method `forward_messages()`**:**

Usage:

```
TelegramClient$forward_messages(
  entity,
  messages,
  from_peer = NULL,
  background = NULL,
  with_my_score = NULL,
  silent = NULL,
  as_album = NULL,
  schedule = NULL,
  drop_author = NULL,
  drop_media_captions = NULL
)
```

Method `edit_message()`**:**

Usage:

```
TelegramClient$edit_message(
  entity,
  message = NULL,
  text = NULL,
  parse_mode = NULL,
  attributes = NULL,
  formatting_entities = NULL,
  link_preview = TRUE,
  file = NULL,
  thumb = NULL,
  force_document = FALSE,
  buttons = NULL,
  supports_streaming = FALSE,
  schedule = NULL
)
```

Method `delete_messages()`**:**

Usage:

```
TelegramClient$delete_messages(entity, message_ids, revoke = TRUE)
```

Method `send_read_acknowledge()`**:**

Usage:

```
TelegramClient$send_read_acknowledge(
  entity,
  message = NULL,
  max_id = NULL,
  clear_mentions = FALSE,
  clear_reactions = FALSE
)
```

Method `pin_message()`**:**

Usage:

```
TelegramClient$pin_message(entity, message, notify = FALSE, pm_oneside = FALSE)
```

Method `unpin_message()`**:**

Usage:

```
TelegramClient$unpin_message(entity, message = NULL, notify = FALSE)
```

Method `get_comment_data()`**:**

Usage:

```
TelegramClient$get_comment_data(entity, message)
```

Method `pin_internal()`**:**

Usage:

```
TelegramClient$pin_internal(
  entity,
  message,
  unpin,
  notify = FALSE,
  pm_oneside = FALSE
)
```

Method `send_file()`**:**

Usage:

```
TelegramClient$send_file(
  entity,
  file,
  caption = NULL,
  force_document = FALSE,
  file_size = NULL,
  progress_callback = NULL,
  ...
)
```

Method `upload_file()`**:**

Usage:

```
TelegramClient$upload_file(
  file,
  part_size_kb = NULL,
  file_size = NULL,
  progress_callback = NULL,
  ...
)
```

Method `file_to_media()`**:**

Usage:

```
TelegramClient$file_to_media(
  file,
  force_document = FALSE,
  file_size = NULL,
  progress_callback = NULL,
  attributes = NULL,
  thumb = NULL,
  allow_cache = TRUE,
  voice_note = FALSE,
  video_note = FALSE,
  supports_streaming = FALSE,
  mime_type = NULL,
  as_image = NULL,
  ttl = NULL,
  nosound_video = NULL
)
```

Method `resize_photo_if_needed()`**:**

Usage:

```
TelegramClient$resize_photo_if_needed(
  file,
  is_image,
  width = 2560,
  height = 2560
)
```

Method `is_image()`**:**

Usage:

```
TelegramClient$is_image(file)
```

Method `get_appropriated_part_size()`**:**

Usage:

```
TelegramClient$get_appropriated_part_size(file_size)
```

Method `invoke()`**:**

Usage:

```
TelegramClient$invoke(request)
```

Method `build_reply_markup()`**:**

Usage:

```
TelegramClient$build_reply_markup(buttons = NULL, inline_only = FALSE)
```

Method `run_until_disconnected()`**:**

Usage:

```
TelegramClient$run_until_disconnected()
```

Method `set_receive_updates()`**:**

Usage:

```
TelegramClient$set_receive_updates(receive_updates)
```

Method `on()`**:**

Usage:

```
TelegramClient$on(event)
```

Method `add_event_handler()`**:**

Usage:

```
TelegramClient$add_event_handler(callback, event = NULL)
```

Method `remove_event_handler()`**:**

Usage:

```
TelegramClient$remove_event_handler(callback, event = NULL)
```

Method `list_event_handlers()`**:**

Usage:

```
TelegramClient$list_event_handlers()
```

Method `catch_up()`**:**

Usage:

```
TelegramClient$catch_up()
```

Method `update_loop()`**:**

Usage:

```
TelegramClient$update_loop()
```

Method `preprocess_updates()`**:**

Usage:

```
TelegramClient$preprocess_updates(updates, users, chats)
```

Method `keepalive_loop()`**:**

Usage:

```
TelegramClient$keepalive_loop()
```

Method `dispatch_update()`**:**

Usage:

```
TelegramClient$dispatch_update(update)
```

Method `handle_auto_reconnect()`**:**

Usage:

```
TelegramClient$handle_auto_reconnect()
```

Method `set_parse_mode()`**:**

Usage:

```
TelegramClient$set_parse_mode(mode)
```

Method `get_parse_mode()`**:**

Usage:

```
TelegramClient$get_parse_mode()
```

Method `sanitize_parse_mode()`**:**

Usage:

```
TelegramClient$sanitize_parse_mode(mode)
```

Method `parse_message_text()`**:**

Usage:

```
TelegramClient$parse_message_text(message, parse_mode = NULL)
```

Method `replace_with_mention()`**:**

Usage:

```
TelegramClient$replace_with_mention(entities, i, user)
```

Method `get_response_message()`**:**

Usage:

```
TelegramClient$get_response_message(request, result, input_chat)
```

Method `call()`**:**

Usage:

```
TelegramClient$call(request, ordered = FALSE, flood_sleep_threshold = NULL)
```

Method `call_internal()`**:**

Usage:

```
TelegramClient$call_internal(
  sender,
  request,
  ordered = FALSE,
  flood_sleep_threshold = NULL
)
```

Method `get_me()`**:**

Usage:

```
TelegramClient$get_me(input_peer = FALSE)
```

Method `self_id()`**:**

Usage:

```
TelegramClient$self_id()
```

Method `is_bot()`**:**

Usage:

```
TelegramClient$is_bot()
```

Method `is_user_authorized()`**:**

Usage:

```
TelegramClient$is_user_authorized()
```

Method `get_entity()`**:**

Usage:

```
TelegramClient$get_entity(entity)
```

Method `get_input_entity()`**:**

Usage:

```
TelegramClient$get_input_entity(peer)
```

Method `getInputEntity()`**:**

Usage:

```
TelegramClient$getInputEntity(peer)
```

Method `getInputPeer()`**:**

Usage:

```
TelegramClient$getInputPeer(peer, allow_self = TRUE, check_hash = TRUE)
```

Method `getInputUser()`**:**

Usage:

```
TelegramClient$getInputUser(entity)
```

Method `getInputChannel()`**:**

Usage:

```
TelegramClient$getInputChannel(entity)
```

Method `getInputMessage()`**:**

Usage:

```
TelegramClient$getInputMessage(message)
```

Method `getInputMedia()`**:**

Usage:

```
TelegramClient$getInputMedia(
  media,
  is_photo = FALSE,
  attributes = NULL,
  force_document = FALSE,
  file_size = NULL,
  progress_callback = NULL
)
```

Method `getInputDocument()`**:**

Usage:

```
TelegramClient$getInputDocument(document)
```

Method `getInputPhoto()`**:**

Usage:

```
TelegramClient$getInputPhoto(photo)
```

Method `getInputChatPhoto()`**:**

Usage:

```
TelegramClient$getInputChatPhoto(photo)
```

Method `getInputGroupCall()`**:**

Usage:

```
TelegramClient$getInputGroupCall(call)
```

Method `get_peer()`**:**

Usage:

```
TelegramClient$get_peer(peer)
```

Method `get_peer_id()`**:**

Usage:

```
TelegramClient$get_peer_id(peer, add_mark = TRUE)
```

Method `get_entity_from_string()`**:**

Usage:

```
TelegramClient$get_entity_from_string(string)
```

Method `get_input_dialog()`**:**

Usage:

```
TelegramClient$get_input_dialog(dialog)
```

Method `get_input_notify()`**:**

Usage:

```
TelegramClient$get_input_notify(notify)
```

Method `download_files_parallel()`**:**

Usage:

```
TelegramClient$download_files_parallel(
  items,
  output_dir = NULL,
  workers = 4L,
  stagger_ms = 500L
)
```

Method `new()`**:**

Usage:

```
TelegramClient$new(...)
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
TelegramClient$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

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

## End(Not run)
```

update_profile *Update your name and/or bio*

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

TelegramClient, 23

update_profile, 44
update_profile_photo, 44
update_username, 45