

# Package ‘atrrr’

June 6, 2026

**Title** Wrapper for the 'AT' Protocol Behind 'Bluesky'

**Version** 0.2.0

**Description** Wraps the 'AT' Protocol (Authenticated Transfer Protocol) behind 'Bluesky' <<https://bsky.social>>. Functions can be used for, among others, retrieving posts and followers from the network or posting content.

**License** MIT + file LICENSE

**Encoding** UTF-8

**Imports** cli, glue, httr2 (>= 1.0.0), methods, purrr, rlang, stringr, snakecase, tibble, utils, lifecycle

**Depends** R (>= 4.2.0)

**LazyData** true

**Suggests** askpass, av, curl, dplyr, forcats, ggplot2, ggraph, igraph, jsonlite, libipldr, magick, mime, knitr, rmarkdown, testthat (>= 3.0.0), tidygraph, webshot2

**Config/testthat/edition** 3

**VignetteBuilder** knitr

**URL** <https://jbgruber.github.io/atrrr/>,  
<https://github.com/JBGruber/atrrr>

**BugReports** <https://github.com/JBGruber/atrrr/issues>

**Config/roxygen2/version** 8.0.0

**NeedsCompilation** no

**Author** Johannes B. Gruber [aut, cre] (ORCID:  
<<https://orcid.org/0000-0001-9177-1772>>),  
Benjamin Guinaudeau [aut, ctb] (ORCID:  
<<https://orcid.org/0000-0001-7206-6875>>),  
Fabio Votta [aut, ctb] (ORCID: <<https://orcid.org/0000-0002-3143-5942>>),  
Nicholas Erskine [ctb]

**Maintainer** Johannes B. Gruber <[JohannesB.Grubergmail.com](mailto:JohannesB.Grubergmail.com)>

**Repository** CRAN

**Date/Publication** 2026-06-06 11:20:02 UTC

## Contents

auth	2
convert_http_to_at	4
fetch_preview	5
follow	5
get_actor_likes	6
get_feed	7
get_feeds_created_by	8
get_feed_likes	9
get_followers	10
get_likes	11
get_list	12
get_own_timeline	14
get_replies	15
get_skeets_authored_by	15
get_starter_pack	16
get_thread	17
get_user_info	18
like_skeet	19
list_chats	20
list_lexicons	21
post	21
post_thread	23
print.bsky_token	24
search_feed	25
search_post	26
search_user	28
skeet_shot	29
stream_firehose	30
<b>Index</b>	<b>32</b>

---

auth	<i>Authenticate for the API</i>
------	---------------------------------

---

### Description

Run authentication for a network using the AT protocol (e.g., 'Blue Sky') and save the token permanently.

### Usage

```
auth(
  user,
  password,
  domain = "https://bsky.app/",
  verbose = TRUE,
```

```

    overwrite = FALSE,
    token = NULL
  )

```

### Arguments

user	Your user handle (e.g. benguinaudeau.bsky.social).
password	Your app password (usually created on <a href="https://bsky.app/settings/app-passwords">https://bsky.app/settings/app-passwords</a> ).
domain	For now <a href="https://bsky.app/">https://bsky.app/</a> , but could change in the future.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
overwrite	If TRUE, overwrites old token without asking for confirmation.
token	(Stale) token object. Usually you don't need to use this. But if you manage your own tokens and they get stale, you can use this parameter and request a fresh token.

### Details

After requesting the token, it is saved in the location returned by `file.path(tools::R_user_dir("atrrr", "cache"), Sys.getenv("BSKY_TOKEN", unset = "token.rds"))`. If you have multiple tokens, you can use `Sys.setenv(BSKY_TOKEN = "filename.rds")` to save/load the token with a different name.

### Value

An authentication token (invisible)

### Examples

```

## Not run:
# request a token
auth() # this will guide you through all steps

# the token is stored in the location returned by this command
file.path(tools::R_user_dir("atrrr", "cache"),
          Sys.getenv("BSKY_TOKEN", unset = "token.rds"))

# to use a different than the default file name for the token, set BSKY_TOKEN
Sys.setenv(BSKY_TOKEN = "identity-2.rds")

# now either rename your token file or request a new token
auth()

# the cache now contains two tokens
list.files(tools::R_user_dir("atrrr", "cache"))

# functions that interact with the API also take a .token argument with the
# path. For example:
tok_path <- file.path(tools::R_user_dir("atrrr", "cache"), "identity-2.rds")

```

```

get_skeets_authored_by(actor = "benguinaudeau.bsky.social", parse = TRUE,
                       .token = tok_path)

## End(Not run)

```

---

convert\_http\_to\_at      *Converts between http URL and AT URI*

---

### Description

Converts between http URL and AT URI

### Usage

```

convert_http_to_at(link, .token = NULL)

convert_at_to_http(link)

```

### Arguments

link	either an AT or HTTP link.
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

### Details

The AT protocol uses a different scheme to link to posts, user, feeds etc. Instead of the common https:// link format, internally a links starting with at:// are used (see <https://atproto.com/specs/at-uri-scheme> for details). The functions convert links from the HTTP to the AT format, or the other way around. This is useful if you want to use a link in a browser.

### Value

either an AT or HTTP link

### Examples

```

## Not run:
convert_http_to_at("https://bsky.app/profile/benguinaudeau.bsky.social/post/3kbi5v7oncq25")
convert_at_to_http("at://did:plc:vuvsifrusnjsys7mhhpk662u/app.bsky.feed.post/3kbi5v7oncq25")

## End(Not run)

```

---

fetch_preview	<i>Fetch link preview</i>
---------------	---------------------------

---

**Description**

Fetch link preview

**Usage**

```
fetch_preview(uri)
```

**Arguments**

uri                   URL or URI to fetch preview for.

**Value**

list structured for use as preview\_card

**Examples**

```
## Not run:
wiki_preview <- fetch_preview("https://en.wikipedia.org/wiki/AT_Protocol")
post_skeet("Do you know the AT Protocol?", preview_card = wiki_preview)

## End(Not run)
```

---

follow	<i>Un/Follow an account</i>
--------	-----------------------------

---

**Description**

Un/Follow an account

**Usage**

```
follow(actor, verbose = NULL, .token = NULL)

unfollow(actor, verbose = NULL, .token = NULL)
```

**Arguments**

actor                   User handle to follow or unfollow

verbose                 Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with `Sys.setenv(ATR_VERBOSE = FALSE)`.

.token                  If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Details**

You can only unfollow accounts which you also followed through the API/the package.

**Value**

list with URI and CID of the record (invisible).

**Examples**

```
## Not run:
# follow our test account
follow("atpr.bsky.social")

# unfollow our test account
unfollow("atpr.bsky.social")

## End(Not run)
```

---

get_actor_likes	<i>Get likes of a user</i>
-----------------	----------------------------

---

**Description**

Get likes of a user

**Usage**

```
get_actor_likes(
  actor,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

actor	user handle to retrieve likes for.
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.

verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame (or nested list) of likes/reposts

**Examples**

```
## Not run:
get_actor_likes("jbgruber.bsky.social")

## End(Not run)
```

---

get_feed	<i>Get the skeets from a specific feed</i>
----------	--

---

**Description**

Get the skeets that would be shown when you open the given feed

**Usage**

```
get_feed(
  feed_url,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

feed_url	The url of the requested feed
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .

.token            If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

### Value

a data frame (or nested list) of posts

### Examples

```
## Not run:
# use the URL of a feed
get_feed("https://bsky.app/profile/did:plc:2zcfjzyocp6kapg6jc4eacok/feed/aaaekvqc3gzg")

# or search for a feed by name
res <- search_feed("#rstats")
get_feed(res$uri[1])

## End(Not run)
```

---

get\_feeds\_created\_by    *A view of the feed created by an actor.*

---

### Description

A view of the feed created by an actor.

### Usage

```
get_feeds_created_by(
  actor,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

### Arguments

actor	user handle to retrieve feed from
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.

verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame (or nested list) of feeds

**Examples**

```
## Not run:
feed <- get_feeds_created_by("andrew.heiss.phd")

## End(Not run)
```

---

get_feed_likes	<i>Get likes of a feed</i>
----------------	----------------------------

---

**Description**

Get likes of a feed

**Usage**

```
get_feed_likes(
  feed_url,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

feed_url	the URL of a feed for which to retrieve who liked it.
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .

.token            If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

### Value

a data frame (or nested list) of likes/reposts

### Examples

```
## Not run:
# use the URL of a feed
get_feed_likes("https://bsky.app/profile/did:plc:2zcfjzyocp6kapg6jc4eacok/feed/aaeckvqc3gzg")

# or search for a feed by name
res <- search_feed("#rstats")
get_feed_likes(res$uri[1])

## End(Not run)
```

---

get\_followers

*Get followers and follows of an actor*

---

### Description

Get followers and follows of an actor

### Usage

```
get_followers(
  actor,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

```
get_follows(
  actor,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

actor	user handle to look up followers for.
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
. token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame (or nested list) of found actors.

**Examples**

```
## Not run:
get_followers("benguinaudeau.bsky.social")

# get first page of results
follows_df <- get_follows("favstats.eu", limit = 25L)

# continue same search, starting from the next match
follows_df2 <- get_follows("favstats.eu", limit = 25L,
                          cursor = attr(follows_df, "last_cursor"))

## End(Not run)
```

---

get\_likes

*Get likes/reposts of a skeet*

---

**Description**

Get likes/reposts of a skeet

**Usage**

```
get_likes(
  post_url,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
```

```

    verbose = NULL,
    .token = NULL
  )

  get_reposts(
    post_url,
    limit = 25L,
    cursor = NULL,
    parse = TRUE,
    verbose = NULL,
    .token = NULL
  )

```

### Arguments

post_url	the URL of a skeet for which to retrieve who liked/reposted it.
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

### Value

a data frame (or nested list) of likes/reposts

### Examples

```

## Not run:
get_likes("https://bsky.app/profile/jbgruber.bsky.social/post/3kbi55xm6u62v")
get_reposts("https://bsky.app/profile/jbgruber.bsky.social/post/3kbi55xm6u62v")

## End(Not run)

```

---

get\_list

*Get List*

---

### Description

Get a feed of recent posts from a list (posts and reposts from any actors on the list).

**Usage**

```
get_list(
  list,
  limit = 25,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

```
get_list_feed(
  list,
  limit = 25,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

list	The url of the requested list
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame (or nested list) of posts

**Examples**

```
## Not run:
# use the URL of a list to get users on the list
get_list("https://bsky.app/profile/smachlis.bsky.social/lists/317o5d7b7n12q")

# or the feed of that list
get_list_feed("https://bsky.app/profile/smachlis.bsky.social/lists/317o5d7b7n12q")

## End(Not run)
```

---

get_own_timeline	<i>Get your own timeline</i>
------------------	------------------------------

---

## Description

Get the posts that would be shown when you open the Bluesky app or website.

## Usage

```
get_own_timeline(
  algorithm = NULL,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

## Arguments

algorithm	algorithm used to sort the posts
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

## Value

a data frame (or nested list) of posts

## Examples

```
## Not run:
get_own_timeline()
get_own_timeline(algorithm = "reverse-chronological")

## End(Not run)
```

---

get_replies	<i>Get all replies</i>
-------------	------------------------

---

**Description**

Get all replies and replies on replies of a skeet.

**Usage**

```
get_replies(post_url, .token = NULL)
```

**Arguments**

post_url	the URL of a skeet.
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame of skeets

**Examples**

```
## Not run:  
get_replies("https://bsky.app/profile/jbgruber.bsky.social/post/3kbi57u4sys2l")  
  
## End(Not run)
```

---

get_skeets_authored_by	<i>A view of an actor's skeets.</i>
------------------------	-------------------------------------

---

**Description**

A view of an actor's skeets.

**Usage**

```
get_skeets_authored_by(  
  actor,  
  limit = 25L,  
  filter = NULL,  
  cursor = NULL,  
  parse = TRUE,  
  verbose = NULL,  
  .token = NULL  
)
```

**Arguments**

actor	user handle to retrieve feed for.
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
filter	get only certain post/repost types. Possible values "posts_with_replies", "posts_no_replies", "posts_with_media", and "posts_and_author_threads".
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
. token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame (or nested list) of posts

**Examples**

```
## Not run:
andrews_posts <- get_skeets_authored_by("andrew.heiss.phd")

## End(Not run)
```

---

get\_starter\_pack      *Get Starter Pack*

---

**Description**

Get all info about a starter pack and the users on it.

**Usage**

```
get_starter_pack(starter_pack, parse = TRUE, .token = NULL)
```

```
get_actor_starter_packs(actor, limit = NULL, cursor = NULL, .token = NULL)
```

**Arguments**

starter_pack	the URL of a starter pack
parse	Parse the results or return the original nested object sent by the server.
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.
actor	user handle of account to query for starter packs.
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).

**Value**

a data frame of users and list info

**Examples**

```
## Not run:
get_starter_pack("https://bsky.app/starter-pack/sof14g1l.bsky.social/3lbc4bqetfp22")

## End(Not run)
```

---

get_thread	<i>Get all skeets in a thread</i>
------------	-----------------------------------

---

**Description**

Retrieve all skeets in a thread (all replies to an original skeet by any author). It does not matter if you use the original skeet or any reply as post\_url.

**Usage**

```
get_thread(post_url, parse = TRUE, .token = NULL)
```

**Arguments**

post_url	the URL of any skeet in a thread.
parse	Parse the results or return the original nested object sent by the server.
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame of skeets

## Examples

```
## Not run:  
get_thread("https://bsky.app/profile/jbgruber.bsky.social/post/3kbi57u4sys21")  
  
## End(Not run)
```

---

get_user_info	<i>Query profile of an actor or actors</i>
---------------	--

---

## Description

Query profile of an actor or actors

## Usage

```
get_user_info(actor, parse = TRUE, .token = NULL)
```

## Arguments

actor	user handle(s) to get information for.
parse	Parse the results or return the original nested object sent by the server.
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

## Details

Note that if you query several actors at once, profiles that do not (longer) exist will be missing from the results. If you query and individual actors and the profile is missing, the API will return an error and a message like "Account has been suspended".

## Value

a data frame (or nested list) of found actors.

## Examples

```
## Not run:  
rstats_user <- search_user("rstats", limit = 2L)  
get_user_info(rstats_user$handle)  
  
## End(Not run)
```

---

like_skeet	<i>Like a skeet</i>
------------	---------------------

---

## Description

Like a skeet

## Usage

```
like_skeet(post_url, verbose = NULL, .token = NULL)
```

```
like_post(post_url, verbose = NULL, .token = NULL)
```

## Arguments

post_url	URL or URI of post to delete.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

## Value

invisible record information from the API

## Examples

```
## Not run:  
# like a post  
like_skeet("https://bsky.app/profile/jbgruber.bsky.social/post/3lcmymlgxwa2t")  
  
# or feed in the result of some search  
johannes_posts <- get_skeets_authored_by("jbgruber.bsky.social")  
like_skeet(johannes_posts$uri)  
  
## End(Not run)
```

---

`list_chats`*Interact with Bluesky Direct Messages*

---

**Description**

Interact with Bluesky Direct Messages

**Usage**

```
list_chats(  
  limit = NULL,  
  cursor = NULL,  
  unread = FALSE,  
  status = NULL,  
  parse = TRUE,  
  .token = NULL  
)
```

```
get_user_chat(actor, parse = TRUE, .token = NULL)
```

```
check_user_chat_available(actor, parse = TRUE, .token = NULL)
```

```
send_chat_message(chat_id, text, .token = NULL)
```

**Arguments**

<code>limit</code>	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
<code>cursor</code>	Cursor for pagination (to pick up an old search).
<code>unread</code>	TRUE/FALSE only show chats with unread messages.
<code>status</code>	request or accepted.
<code>parse</code>	Parse the results or return the original nested object sent by the server.
<code>.token</code>	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.
<code>actor</code>	user DID or handle to get chats for.
<code>chat_id</code>	id of the chat. You can get it with <code>list_chats</code> , <code>get_user_chat</code> , or <code>check_user_chat_available</code> .
<code>text</code>	text to send to other user.

**Value**

different objects.

**Examples**

```
## Not run:
list_chats(readState = "read")

## End(Not run)
```

---

list_lexicons	<i>AT Protocol Lexicons</i>
---------------	-----------------------------

---

**Description**

Available lexicons for the AT Protocol (Authenticated Transfer Protocol)

**Usage**

```
list_lexicons
```

**Format**

```
list_lexicons:
A list with all lexicons available for the AT protocols.
names Name of the lexicon
values path relative to https://github.com/bluesky-social/atproto/tree/main/lexicons
```

**Source**

<https://github.com/bluesky-social/atproto>

---

post	<i>Post a skeet</i>
------	---------------------

---

**Description**

Post a skeet

**Usage**

```
post(
  text,
  in_reply_to = NULL,
  quote = NULL,
  image = NULL,
  image_alt = NULL,
  video = NULL,
  link = NULL,
```

```

    created_at = Sys.time(),
    labels = NULL,
    langs = NULL,
    tags = NULL,
    preview_card = TRUE,
    verbose = NULL,
    .token = NULL,
    .reply = NULL
)

post_skeet(
  text,
  in_reply_to = NULL,
  quote = NULL,
  image = NULL,
  image_alt = NULL,
  video = NULL,
  link = NULL,
  created_at = Sys.time(),
  labels = NULL,
  langs = NULL,
  tags = NULL,
  preview_card = TRUE,
  verbose = NULL,
  .token = NULL,
  .reply = NULL
)

delete_skeet(post_url, verbose = NULL, .token = NULL)

delete_post(post_url, verbose = NULL, .token = NULL)

```

### Arguments

text	Text to post
in_reply_to	URL or URI of a skeet this should reply to.
quote	URL or URI of a skeet this should quote.
image, video	path to an image or video to post.
image_alt	alt text for the image.
link	instead of adding a link in text (gets parsed automatically), it's also possible to add a link directly (and save some characters).
created_at	time stamp of the post.
labels	can be used to label a post, for example "!no-unauthenticated", "porn", "sexual", "nudity", or "graphic-media".
langs	indicates human language(s) (up to 3) of post's primary text content.
tags	additional hashtags, in addition to any included in post text and facets.

preview_card	logical. Display a preview card for links included in the text or link (if images or videos are included, they take precedence). alternatively, fetch a card with <code>fetch_preview()</code> and supply the object here.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.
.reply	a pre-built reply list with root and parent elements (each containing <code>uri</code> and <code>cid</code> ). Used for <code>post_thread()</code> not really important for users.
post_url	URL or URI of post to delete.

**Value**

list of the URI and CID of the post (invisible)

**Examples**

```
## Not run:
post("Hello from #rstats with {atrrr}")

## End(Not run)
```

---

post_thread	<i>Post a thread</i>
-------------	----------------------

---

**Description**

Post a thread

**Usage**

```
post_thread(
  texts,
  images = NULL,
  image_alts = NULL,
  thread_df = NULL,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

texts	a vector of skeet (post) texts
images	paths to images to be included in each post. This may be a character vector, or a list of character vectors if multiple images per post are required.

<code>image_alts</code>	alt texts for the images to be included in each post. If <code>images</code> is a list of character vectors, this should also be a list of character vectors and have the same shape.
<code>thread_df</code>	instead of defining <code>texts</code> , <code>images</code> and <code>image_alts</code> , you can also create a data frame with the information in columns <code>text</code> , <code>image</code> , and <code>image_alt</code> .
<code>verbose</code>	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
<code>.token</code>	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

list of the URIs and CIDs of the posts (invisible)

**Examples**

```
## Not run:
# post three messages in a thread
thread <- post_thread(c("Post 1", "Post 2", "Post 3"))

# delete the thread
delete_post(thread$uri)

## End(Not run)
```

---

`print.bsky_token`      *Print token*

---

**Description**

Print a AT token

**Usage**

```
## S3 method for class 'bsky_token'
print(x, ...)
```

**Arguments**

<code>x</code>	An object of class <code>bsky_token</code>
<code>...</code>	not used.

**Value**

No return value, called to print the token to screen

---

search_feed	<i>Search a specific feed</i>
-------------	-------------------------------

---

### Description

Search the feed named after a given query

### Usage

```
search_feed(  
  query,  
  limit = 25L,  
  cursor = NULL,  
  parse = TRUE,  
  verbose = NULL,  
  .token = NULL  
)
```

### Arguments

query	The term to be searched
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
.token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

### Value

a data frame (or nested list) of posts

### Examples

```
## Not run:  
search_feed("rstats")  
  
## End(Not run)
```

---

search\_post

*Search Posts*

---

## Description

Search Posts

## Usage

```
search_post(  
  q,  
  limit = 100L,  
  sort = NULL,  
  since = NULL,  
  until = NULL,  
  mentions = NULL,  
  author = NULL,  
  lang = NULL,  
  domain = NULL,  
  url = NULL,  
  tag = NULL,  
  parse = TRUE,  
  verbose = NULL,  
  .token = NULL  
)
```

```
search_skeet(  
  q,  
  limit = 100L,  
  sort = NULL,  
  since = NULL,  
  until = NULL,  
  mentions = NULL,  
  author = NULL,  
  lang = NULL,  
  domain = NULL,  
  url = NULL,  
  tag = NULL,  
  parse = TRUE,  
  verbose = NULL,  
  .token = NULL  
)
```

## Arguments

q search query. See Details.

limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
sort	string. Specifies the ranking order of results. Possible values are "top" or "latest". Defaults to "latest".
since	string. Filter results for posts after the specified datetime (inclusive). Can be a date or datetime object or a string that can be parsed either.
until	string. Filter results for posts before the specified datetime (not inclusive). Can be a date or datetime object or a string that can be parsed either.
mentions	string. Filter to posts that mention the given account. Only matches rich-text facet mentions.
author	string. Filter to posts authored by the specified account.
lang	string. Filter results to posts in the specified language. Language detection is expected to use the post's language field, though the server may override detection.
domain	string. Filter results to posts containing URLs (links or embeds) pointing to the specified domain. Hostname normalization may apply.
url	string. Filter results to posts containing links or embeds matching the specified URL. URL normalization or fuzzy matching may apply.
tag	string. Filter results to posts containing the specified tag (hashtag). Do not include the hash (#) prefix. Multiple tags can be specified, with results matching all specified tags (logical AND).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
. token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

## Details

The [API docs](#) claim that Lucene query syntax is supported (Boolean operators and brackets for complex queries). But only a small subset is actually implemented:

- Whitespace is treated as implicit AND, so all words in a query must occur, but the word order and proximity are ignored.
- Double quotes indicate exact phrases.
- `from:<handle>` will filter to results from that account.
- `-` excludes terms.
- OR and parentheses for grouping (e.g. `(rstats OR rstat) bluesky`) are **not** supported and are treated literally (i.e. will only find posts with parentheses and the word OR in it).

Note that matches can occur anywhere in the skeet, not just the text. For example, a term can be in the link preview, or alt text of an image.

**Value**

a data frame (or nested list) of posts

**Examples**

```
## Not run:
search_post("rstats")
# finds post with the hashtag rstats AND the word Bluesky somewhere in the
# skeet (ignoring capitalisaion)
search_post("#rstats Bluesky")

# search for the exact phrase "new #rstats package"
search_post("\"new #rstats package\"")
# Use single quotes so you do not need to escape double quotes
search_post('"new #rstats package"')

# only search for skeets from one user
search_post("from:jbgruber.bsky.social #rstats")

# narrow down the search with more parameters
search_post("{atrrr}",
            sort = "top",
            since = "2024-12-05",
            until = "2024-12-07 10:00:00",
            mentions = NULL,
            author = "jbgruber.bsky.social",
            domain = "jbgruber.github.io",
            url = "https://jbgruber.github.io/atrrr",
            tag = "rstats")

## End(Not run)
```

---

search\_user

*Find users (profiles) matching search criteria.*

---

**Description**

Find users (profiles) matching search criteria.

**Usage**

```
search_user(
  query,
  limit = 25L,
  cursor = NULL,
  parse = TRUE,
  verbose = NULL,
  .token = NULL
)
```

**Arguments**

query	The search query. Searches in user names and descriptions or <b>exact</b> matches in user handles (including the <i>.bsky.social</i> part).
limit	Maximum number of records to return. For queries with more than 100 results, pagination is used automatically (one request per 100 results). The function stops when the limit is reached, but you will usually get a few items more than requested.
cursor	Cursor for pagination (to pick up an old search).
parse	Parse the results or return the original nested object sent by the server.
verbose	Whether to print status messages to the Console (TRUE/FALSE). Package default (when NULL) is to have status messages. Can be changed with <code>Sys.setenv(ATR_VERBOSE = FALSE)</code> .
. token	If you manage your own tokens, you can supply it here. Usually NULL is OK and will automatically load or guide you to generate a token.

**Value**

a data frame (or nested list) of found actors.

**Examples**

```
## Not run:
search_user("benguinaudeau.bsky.social")
search_user("Blog: favstats.eu")
search_user("JBGruber")
search_user("@UvA_ASCoR")
search_user("rstats", limit = 1000L)

## End(Not run)
```

---

skeet\_shot

*Take high quality screenshots of skeets*


---

**Description**

Take high quality screenshots of skeets

**Usage**

```
skeet_shot(x, file = NULL, delay = 1, ...)
```

**Arguments**

x	a vector of URLs or URIs.
file	output file name. Defaults to the skeet id.
delay	time to wait for content to load. Can make sense to increase if not everything is loaded in the screenshot yet.
...	passed on to <a href="#">webshot</a> .

**Value**

path to file

**Examples**

```
## Not run:
df <- atrrr::search_post("rstats")
skeet_shot(df$uri[1:2])

## End(Not run)
```

---

stream_firehose	<i>Stream from the Bluesky Firehose</i>
-----------------	---

---

**Description****[Experimental]**

Bluesky offers a stream of all events created, edited and deleted on the platform and the wider AT protocol using network. This function connects to it and decrypts the stream. See examples for how to provide a function that handles the firehose stream.

**Usage**

```
stream_firehose(fun, timeout = 30)
```

**Arguments**

fun	a function that takes events from the Bluesky Firehose and processes them. See examples.
timeout	how long to stream from the Bluesky Firehose. Can be Inf to keep streaming until interrupted.

**Value**

nothing itself, used to run fun

**Examples**

```
## Not run:
# an example function that saves events as RDS files
stream2rds <- function(events) {
  for (event in events) {
    commit <- event$commit
    # ignore events without a commit message
    if (!is.null(commit)) {
      saveRDS(event, paste0(event$commit, ".rds"), compress = FALSE)
    }
  }
}
```

```
}
stream_firehose(stream2rds, timeout = 15)

# an example function that saves events as JSON files
stream2json <- function(events) {
  for (event in events) {
    commit <- event$commit
    # ignore events without a commit message
    if (!is.null(commit)) {
      jsonlite::write_json(event, file(paste0(event$commit, ".json")), pretty = TRUE)
    }
  }
}
stream_firehose(stream2json, timeout = 15)

# an example function that prints posts to the Console
print_stream <- function(events) {
  for (event in events) {
    commit <- event$commit
    # ignore events without a commit message
    if (is.null(commit)) next
    # decode actual block data which contains text
    event$data <- libipldr::decode_car(event$blocks)$blocks
    message <- purrr::map(event$data, "text", .default = NULL) |>
      purrr::compact()
    if (length(message) == 0L) next
    print(cli::boxx(message, background_col = sample(colors(), 1L)))
  }
}
stream_firehose(print_stream, timeout = 15)

## End(Not run)
```

# Index

## \* datasets

- [list\\_lexicons](#), [21](#)
- [auth](#), [2](#)
- [check\\_user\\_chat\\_available](#) ([list\\_chats](#)), [20](#)
- [convert\\_at\\_to\\_http](#) ([convert\\_http\\_to\\_at](#)), [4](#)
- [convert\\_http\\_to\\_at](#), [4](#)
- [delete\\_post](#) ([post](#)), [21](#)
- [delete\\_skeet](#) ([post](#)), [21](#)
- [fetch\\_preview](#), [5](#)
- [fetch\\_preview\(\)](#), [23](#)
- [follow](#), [5](#)
- [get\\_actor\\_likes](#), [6](#)
- [get\\_actor\\_starter\\_packs](#) ([get\\_starter\\_pack](#)), [16](#)
- [get\\_feed](#), [7](#)
- [get\\_feed\\_likes](#), [9](#)
- [get\\_feeds\\_created\\_by](#), [8](#)
- [get\\_followers](#), [10](#)
- [get\\_follows](#) ([get\\_followers](#)), [10](#)
- [get\\_likes](#), [11](#)
- [get\\_list](#), [12](#)
- [get\\_list\\_feed](#) ([get\\_list](#)), [12](#)
- [get\\_own\\_timeline](#), [14](#)
- [get\\_replies](#), [15](#)
- [get\\_reposts](#) ([get\\_likes](#)), [11](#)
- [get\\_skeets\\_authored\\_by](#), [15](#)
- [get\\_starter\\_pack](#), [16](#)
- [get\\_thread](#), [17](#)
- [get\\_user\\_chat](#) ([list\\_chats](#)), [20](#)
- [get\\_user\\_info](#), [18](#)
- [like\\_post](#) ([like\\_skeet](#)), [19](#)
- [like\\_skeet](#), [19](#)
- [list\\_chats](#), [20](#)

- [list\\_lexicons](#), [21](#)
- [post](#), [21](#)
- [post\\_skeet](#) ([post](#)), [21](#)
- [post\\_thread](#), [23](#)
- [post\\_thread\(\)](#), [23](#)
- [print.bsky\\_token](#), [24](#)
- [search\\_feed](#), [25](#)
- [search\\_post](#), [26](#)
- [search\\_skeet](#) ([search\\_post](#)), [26](#)
- [search\\_user](#), [28](#)
- [send\\_chat\\_message](#) ([list\\_chats](#)), [20](#)
- [skeet\\_shot](#), [29](#)
- [stream\\_firehose](#), [30](#)
- [unfollow](#) ([follow](#)), [5](#)
- [webshot](#), [29](#)