Package 'atrrr'

March 5, 2024

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

Version 0.0.3

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

RoxygenNote 7.3.1

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

Depends R (>= 4.1)

LazyData true

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

Config/testthat/edition 3

VignetteBuilder knitr

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

NeedsCompilation no

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

Maintainer Johannes B. Gruber <JohannesB.Gruber@gmail.com>

Repository CRAN

Date/Publication 2024-03-05 15:40:02 UTC

R topics documented:

`auth` *Authenticate for the API*

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 https://bsky.app/settings/app-passwords).
domain	For now https://bsky.app/, 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 Sys.setenv(ATR_VERBOSE = FALSE).
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 betweet http URL and AT URI*

Description

Converts betweet 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, useser, 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:vuvsifrusnjsys7mhkpk662u/app.bsky.feed.post/3kbi5v7oncq25")

## End(Not run)
```

follow *Un/Follow an account*

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_feed *Get the skeets from a specific feed*

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 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.

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/aaaeckvqc3gzg")

# 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 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.

Value

a data frame (or nested list) of feeds

Examples

```
## Not run:
feed <- get_feeds_created_by("profmusgrave.bsky.social")

## End(Not run)
```

`get_feed_likes` *Get likes of a feed*

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 `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.

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/aaaeckvqc3gzg")

# 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 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.

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.bsky.social", limit = 25L)

# continue same search, starting from the next match
follows_df2 <- get_follows("favstats.bsky.social", 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 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.

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_own_timeline *Get your own timeline*

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 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.

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 *Get all replies*

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 *A view of an actor's skeets.*

Description

A view of an actor's skeets.

Usage

```
get_skeets_authored_by(
  actor,
  limit = 25L,
  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.
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 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.

Value

a data frame (or nested list) of posts

Examples

```
## Not run:
feed <- get_skeets_authored_by("profmusgrave.bsky.social")

## End(Not run)
```

get_thread *Get all skeets in a thread*

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, .token = NULL)
```

Arguments

`post_url`	the URL of any skeet in a thread.
`.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/3kbi57u4sys2l")

## End(Not run)
```

get_user_info *Query profile of an actor*

Description

Query profile of an actor

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.

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)
```

---

list_lexicons *AT Protocol Lexicons*

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 *Post a skeet*

Description

Post a skeet

Usage

```
post(
  text,
  in_reply_to = NULL,
  quote = NULL,
  image = NULL,
  image_alt = NULL,
  created_at = Sys.time(),
  preview_card = TRUE,
  verbose = NULL,
  .token = NULL
)

post_skeet(
  text,
  in_reply_to = NULL,
  quote = NULL,
  image = NULL,
  image_alt = NULL,
  created_at = Sys.time(),
  preview_card = TRUE,
  verbose = NULL,
  .token = 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	path to an image to post.
image_alt	alt text for the image.
created_at	time stamp of the post.
preview_card	display a preview card for links included in the text (only if image is NULL).
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.
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` *Post a thread*

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
`image_alts`	alt texts for the images to be included in each post
`thread_df`	instead of defining texts, images and image_alts, you can also create a data frame with the information in columns of the same names.
`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.

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

x	An object of class bsky_token
...	not used.

Value

No return value, called to print the token to screen

---

search_feed *Search a specific feed*

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 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.

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, parse = TRUE, verbose = NULL, .token = NULL)

search_skeet(q, limit = 100L, 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.

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 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

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 (does not seem to be working at the moment).

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")

## 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 **exact** matches in user handles (including the *.bsky.social* 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 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.

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, ...)
```

Arguments

x	a vector of URLs or URIs.
file	output file name. Defaults to the skeet id.
...	passed on to webshot.

Value

path to file

Examples

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

## End(Not run)
```

Index