

Package: promoutils (via r-universe)

February 25, 2026

Title Utilities for Promoting rOpenSci on Social Media

Version 0.4.4

Description Utility functions for accessing GitHub and social media data.

License GPL (≥ 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports cli, clipr, dplyr, gh, glue, httr2 ($\geq 1.0.0$), jsonlite, lubridate, magrittr, memoise ($\geq 2.0.1$), pandoc, purrr, readr, rlang, rvest, stringr, tidyr, yaml

Suggests httpuv, testthat ($\geq 3.0.0$)

Config/testthat/edition 3

Config/ropensci/maintainer staff

Depends R ($\geq 4.0.0$)

URL <https://github.com/ropensci-org/promoutils>

BugReports <https://github.com/ropensci-org/promoutils/issues>

Config/pak/sysreqs git make libicu-dev libxml2-dev libssl-dev libx11-dev

Repository <https://ropensci.r-universe.dev>

Date/Publication 2026-02-25 16:13:47 UTC

RemoteUrl <https://github.com/ropensci-org/promoutils>

RemoteRef main

RemoteSha 754bf5c94091be236d01dc8249e3c67810edfdf8

Contents

all_users	3
cw_checkin	3
cw_details	4
cw_event	4
cw_issue	5
cw_slack_hour	5
cw_socials	6
discourse_user	7
forum_mention	7
forum_post	8
forum_resource	8
gh_cache	9
gh_issue_fetch	9
gh_issue_fmt	10
gh_issue_labels	11
gh_masto	12
gh_name	12
gh_user	13
li_auth	13
li_posts_read	14
li_posts_write	15
li_urn_me	16
masto2user	16
masto_user	17
next_date	17
pkgs	18
pkg_authors	19
replace_emoji	19
ro_masto	20
slack_channels	20
slack_cleanup	21
slack_messages	21
slack_posts_write	22
slack_scheduled_list	23
slack_scheduled_rm	23
slack_user	24
slack_users	25
socials_post_issue	25
yaml_extract	26

Index

27

all_users	<i>Fetch GH and Mastodon usernames</i>
-----------	--

Description

Based on a name and a repository, fetch the usernames

Usage

```
all_users(name, owner = "ropensci", pkg)
```

Arguments

name	Character. Full name
owner	Character. Repository owner.
pkg	Character. Repository name (package name).

Value

Data frame of user names: gh_user and masto_user

Examples

```
all_users(name = "Steffi LaZerte", pkg = "weathercan")
```

cw_checkin	<i>Create draft message for checking in with Cohosts</i>
------------	--

Description

The week before, prepare the slides and coworking document, then use this draft text to invite the cohost(s) to review via Slack or Email.

Usage

```
cw_checkin(which = "next", names = NULL, notes_link, slides_link)
```

Arguments

which	Character/Date. "next" to fetch details on the next coworking session, "last" to fetch details on the last scheduled (in future) coworking session, or a Date fetch details for a specific coworking session.
names	Character. Names of cohost if overriding those in the event listing.
notes_link	Character. Link to the Google doc with coworking notes.
slides_link	Character. Link to the Google slides.

cw_details
Fetch details about coworking sessions

Description

Fetch details about coworking sessions

Usage

```
cw_details(which = "next")
```

Arguments

which Character/Date. "next" to fetch details on the next coworking session, "last" to fetch details on the last scheduled (in future) coworking session, or a Date fetch details for a specific coworking session.

Value

Data frame with coworking event details

Examples

```
cw_details()
cw_details("2023-11")
```

cw_event
Create a draft event for coworking

Description

Creates a draft coworking event for the roweb3 website. All details pulled from the coworking todo list issue in `rosadmin/comms`.

Usage

```
cw_event(date, dry_run = FALSE)
```

Arguments

date Character/Date. Date of the coworking event (local)

dry_run Logical. Whether to really create the event or just return the text.

cw_issue	<i>Create Coworking To-Do's on Comms Repo</i>
----------	---

Description

Create an issue listing the coworking todos. If no date or timezone, picks the next appropriate date (first Tuesday in the month following an existing coworking issue) and next appropriate timezone (cycling through America, Europe, and Australia) automatically.

Usage

```

cw_issue(
  date = NULL,
  tz = NULL,
  theme = "XXXX",
  cohost = "XXXX",
  dry_run = FALSE
)

```

Arguments

<code>date</code>	Character. Date of next event (if NULL picks next first Tuesday).
<code>tz</code>	Character. Timezone of next event (if NULL picks next in order).
<code>theme</code>	Character. Name of theme, if unknown, uses XXXX placeholder
<code>cohost</code>	Character. Name of cohost, if unknown, uses XXXX placeholder
<code>dry_run</code>	Logical. Whether to do a dry run (i.e. don't post)

cw_slack_hour	<i>Schedule 1-hour before messages on rOpenSci Slack</i>
---------------	--

Description

Will only work if running between the time that the 1-week message was posted and the start of the coworking.

Usage

```

cw_slack_hour(user = "UNRAUCMTK", dry_run = FALSE)

```

Examples

```

## Not run:
  cw_slack_hour(dry_run = TRUE)

## End(Not run)

```

 cw_socials

Create a draft post for coworking

Description

Creates draft posts for Mastodon and LinkedIn (by opening issues on rosadmin/scheduled_socials) and Slack (by printing the post text and schedule).

Usage

```

cw_socials(
  date,
  who_masto,
  who_slack,
  who_linkedin,
  who_main_masto = "@steffilazerte@fosstodon.org",
  who_main_slack = "<@UNRAUCMTK>",
  who_main_linkedin = "Steffi LaZerte",
  posters_tz = "America/Winnipeg",
  dry_run = FALSE,
  branch = NULL
)

```

Arguments

<code>date</code>	Character/Date. Date of the coworking event (local)
<code>who_masto</code>	Character. The full mastodon handle for the cohost (i.e. XXXX@XXXX.com)
<code>who_slack</code>	Character. The full API Slack id for the cohost (i.e. <@UXXXXXXXX>)
<code>who_linkedin</code>	Character. The full LinkedIn handle for the cohost (i.e. @XXXX)
<code>who_main_masto</code>	Character. The full mastodon handle for the rOpenSci staff organizer.
<code>who_main_slack</code>	Character. The Slack id for the rOpenSci staff organizer (i.e., <@UXXXXXXXX>. Defaults to Steffi's id.
<code>who_main_linkedin</code>	Character. The full LinkedIn handle for the rOpenSci staff organizer.
<code>posters_tz</code>	Character. Timezone of poster. Required for getting the time at which to post Slack messages as these are posted in the local timezone
<code>dry_run</code>	Logical. Whether to do a dry run (i.e. don't post)
<code>branch</code>	Character. Branch name if not on main.

Examples

```
## Not run:
cw_socials("2023-07-04", who_masto = "@cohost@mastodon.org", who_slack = "<UXXXXX>")

## End(Not run)
```

discourse_user	<i>Fetch full name of Discourse user (id)</i>
----------------	---

Description

Using the Discourse username id, return the full name of that discourse user.

Usage

```
discourse_user(user)
```

Arguments

`user_id` Integer. Discourse user id.

Value

Character

Examples

```
# discourse_user(1) # Requires authentication
```

forum_mention	<i>Extract mentions from forum text</i>
---------------	---

Description

Extract mentions from forum text

Usage

```
forum_mention(x)
```

Arguments

`x` Forum text

Value

Character of metions

forum_post	<i>Fetch post text from by topic id</i>
------------	---

Description

Fetch post text from by topic id

Usage

```
forum_post(topic_id)
```

Arguments

x	Topic id
---	----------

Value

HTML of the post

Examples

```
# forum_post(3920) # Needs auth
```

forum_resource	<i>Extract resources from forum text</i>
----------------	--

Description

Extract resources from forum text

Usage

```
forum_resource(x)
```

Arguments

x	Forum text
---	------------

Value

Character vector of resources

Examples

```
# forum_post(3920) |> # Needs auth  
# forum_resource() # > weather0z
```

gh_cache	<i>Create a cached version of the GH api calls</i>
----------	--

Description

Create a cached version of the GH api calls

Usage

```
gh_cache(  
  endpoint,  
  ...,  
  per_page = NULL,  
  .per_page = NULL,  
  .token = NULL,  
  .destfile = NULL,  
  .overwrite = FALSE,  
  .api_url = NULL,  
  .method = "GET",  
  .limit = NULL,  
  .accept = "application/vnd.github.v3+json",  
  .send_headers = NULL,  
  .progress = TRUE,  
  .params = list(),  
  .max_wait = 600,  
  .max_rate = NULL  
)
```

Details

```
memoise::memoise(gh::gh)
```

gh_issue_fetch	<i>Fetch issues from a GH repository</i>
----------------	--

Description

Fetch issues from a GH repository

Usage

```
gh_issue_fetch(  
  state = "open",  
  labels = NULL,  
  since = NULL,  
  owner = "rosadmin",
```

```

  repo = "scheduled_socials",
  issue = NULL,
  verbose = FALSE
)

```

Arguments

<code>state</code>	Character. Which issues to fetch: "open", "closed", "all"
<code>labels</code>	Character vector. Fetch only issues with these labels. (Optional)
<code>since</code>	Character/Date/datetime. Fetch only issues since this date/time. (Optional)
<code>owner</code>	Character. Owner of the repository
<code>repo</code>	Character. Name of the repository (name of the package)
<code>issue</code>	Numeric. Specific Issue number to fetch.
<code>verbose</code>	Logical. Show progress messages.

Value

List of issues

Examples

```
i <- gh_issue_fetch()
```

<code>gh_issue_fmt</code>	<i>Format issues list from GH</i>
---------------------------	-----------------------------------

Description

Format issues list from GH

Usage

```

gh_issue_fmt(
  i,
  which = c("title", "number", "body", "labels", "url", "created", "updated",
           "gh_user_issue")
)

```

Arguments

<code>i</code>	List of issues from <code>gh_issue_fetch()</code>
<code>which</code>	Which fields to include

Value

Issues formatted as a data frame

Examples

```
i <- gh_issue_fetch()
i <- gh_issue_fmt(i, which = "title")
```

gh_issue_labels	<i>Extract information on help-wanted labels for issues</i>
-----------------	---

Description

Extract information on help-wanted labels for issues

Usage

```
gh_issue_labels(
  i,
  labels_help = "(help)|(help wanted)|(help-wanted)|(help_wanted)",
  labels_first = "(good first issue)|(beginner)|(good-first-issue)"
)
```

Arguments

i Data frame of issues from `gh_issue_fmt()`

labels_help Character. Regular expression to match help-wanted labels.

labels_first Character. Regular expression to meatch good-first-issue labels.

Value

data frame with added label details

Examples

```
i <- gh_issue_fetch(owner = "ropensci", repo = "weathercan")
i <- gh_issue_fmt(i)
i <- gh_issue_labels(i)
```

gh_masto	<i>Find Mastodon username from GitHub profile</i>
----------	---

Description

Find Mastodon username from GitHub profile

Usage

```
gh_masto(gh_user)
```

Arguments

gh_user Character. GH username

Value

Character URL to mastodon profile if it exists, NA otherwise.

Examples

```
gh_masto("steffilazerte")
```

gh_name	<i>Fetch full name of GitHub user</i>
---------	---------------------------------------

Description

Fetch full name of GitHub user

Usage

```
gh_name(gh_user)
```

Arguments

gh_user Character. GitHub username/handle

Value

Character, full name of the user or NA

Examples

```
gh_name("steffilazerte")
```

gh_user	<i>Find GH username from repository and full name</i>
---------	---

Description

Look up users of a repository and match to a name. Try with and without initials.

Usage

```
gh_user(name, owner = "ropensci", pkg, .max_rate = NULL)
```

Arguments

name	Character. Full or partial name of the person for whom you want to fetch the GitHub username.
owner	Character. Owner of the repository.
pkg	Character. Repository name (package name).
.max_rate	Numeric. Passed through to <code>gh:gh()</code> .

Value

Data frame with names attempted and usernames found

Examples

```
gh_user(name = "Steffi E. LaZerte", owner = "ropensci", pkg = "weathercan")
gh_user(name = "Steffi", owner = "ropensci", pkg = "weathercan")
```

li_auth	<i>Authorize rOpenSci client with LinkedIn</i>
---------	--

Description

This authorizes the rOpenSci client with **your** credentials (and you must be part of the rOpenSci organization as an admin). Make sure to take note of the 'refresh_token' as that is what you'll add to your .Renviron file for local work, or the GitHub secrets for the comms/scheduled_socials workflow.

Usage

```
li_auth()
```

Details

This function authorizes with a redirect url of "http://localhost:1444/", this *must* be the same as that listed in the LinkedIn Developer App, <https://www.linkedin.com/developers/apps>.

If you retrieve a new token, you will have to put it in the .Renviron and the re-start your R session to continue

This function authorizes with the scopes:

- w_member_social (default)
- w_organization_social (special request)
- r_organization_social (special request)
- r_organization_admin (special request)

Value

httr2 authorization

References

- Refresh tokens API: <https://learn.microsoft.com/en-us/linkedin/shared/authentication/programmatic-refresh-tokens>

Examples

```
## Not run:
# Only run if you need to update the scopes or get a new token (otherwise
# you'll have to replace all your tokens)
t <- li_auth()
t$refresh_token

## End(Not run)
```

li_posts_read	<i>Get a list of recent posts by rOpenSci</i>
---------------	---

Description

Get a list of recent posts by rOpenSci

Usage

```
li_posts_read(author)
```

Arguments

author Character. LinkedIn URN id (e.g., rOpenSci's is "urn:li:organization:77132573")

Value

list of posts

References

- <https://learn.microsoft.com/en-us/linkedin/shared/api-guide/concepts/urns>
- <https://learn.microsoft.com/en-us/linkedin/marketing/integrations/community-management/shares/posts-api>

Examples

```
li_posts_read(ro_urn)
```

li_posts_write	<i>Post to LinkedIn</i>
----------------	-------------------------

Description

Post to LinkedIn

Usage

```
li_posts_write(author, body, dry_run = FALSE)
```

Arguments

author	Character. URN. Either yours (see <code>li_urn_me()</code> or rOpenSci's "urn:li:organization:77132573")
body	Character. The body of the post as you would like it to appear.
dry_run	Logical. TRUE to show what would be sent to the server without actually sending it.

Value

A string of the URN for the post id.

Examples

```
# Dry-run
id <- li_posts_write(
  author = ro_urn, # Post on behalf of rOpenSci
  body = "Testing out the LinkedIn API via R and httr2!",
  dry_run = TRUE)

## Not run:
# Real post
id <- li_posts_write(
```

```

author = ro_urn, # Post on behalf of rOpenSci
body = "Testing out the LinkedIn API via R and httr2!")

## End(Not run)

```

<code>li_urn_me</code>	<i>Fetch your personal URN number</i>
------------------------	---------------------------------------

Description

This is required to post on LinkedIn to your personal account (for rOpenSci, use the organization urn, "urn:li:organization:77132573")

Usage

```
li_urn_me()
```

Value

A string with your URN in the format of "urn:li:person:XXXX"

Examples

```

## Not run:
li_urn_me()

## End(Not run)

```

<code>masto2user</code>	<i>Convert a mastodon user link to handle</i>
-------------------------	---

Description

Convert a mastodon user link to handle

Usage

```
masto2user(x)
```

Arguments

`x` Character. Link to user's profile

Value

Character user handle @user@instance

Examples

```
masto2user("https://fosstodon.org/@steffilazerte")
masto2user("steffi")
masto2user("@steffilazerte@fosstodon.org")
masto2user(NA)
```

masto_user	<i>Fetch Mastodon username</i>
------------	--------------------------------

Description

Using the GH username or the Full name, check rOpenSci author pages and then GitHub for references to the person Mastodon account.

Usage

```
masto_user(gh_user = NULL, name = NULL)
```

Arguments

gh_user	Character. GH user name.
name	Character. Full/Partial name

Value

Character url to Mastodon profile

Examples

```
masto_user("steffilazerte")
```

next_date	<i>Find the next date</i>
-----------	---------------------------

Description

Given a date and a day of the week, Given a date return the next month's first Tuesday

Usage

```
next_date(month, which = "Tues", n = 1)
```

Arguments

<code>month</code>	Character/Date. The current month. Date returned is the next month.
<code>which</code>	Character/Numeric. Which week day to return. Either number or abbreviated English weekday.
<code>n</code>	Numeric. The nth week to return (i.e. the 1st Tuesday if <code>n = 1</code> and <code>which = "Tues"</code>).

Value

A date

Examples

```
# Get the next first Tuesday
next_date("2023-11-01")
next_date("2023-11-30")

# Get the next 3rd Tuesday
next_date("2023-11-01", n = 3)

# Oops
## Not run:
next_date("2023-11-01", n = 5)

## End(Not run)
```

pkgs

Return a data frame of rOpenSci packages

Description

Return a data frame of rOpenSci packages

Usage

```
pkgs(
  url = "https://ropensci.github.io/roregistry/registry.json",
  which = "active",
  return = "sub"
)
```

Arguments

<code>url</code>	Character. Registry url
<code>which</code>	Character. Status of packages to return ("all" or "active")
<code>return</code>	Character. Return a subset ("sub") or all ("all") package fields.

Value

data frame

Examples

```
pkgs()
pkgs(which = "all", return = "all")
```

pkg_authors *Get package author names*

Description

Get package author names

Usage

```
pkg_authors(x, pkgs)
```

Arguments

- x Character. Package name
- pkgs Data frame. Packages returned by `pkgs()`.

Value

Character name of maintainer

replace_emoji *Replace emoji codes with unicode*

Description

Replaces emoji codes like `:tada:` with unicode like `🎉`.

Usage

```
replace_emoji(x)
```

Arguments

- x Character. Text string within which to replace codes

Value

Text string with emoji unicodes

Examples

```
x <- replace_emoji("hi :tada: testing \n\n\n Whow ! \n\n\n :smile:")
x
```

ro_masto	<i>Find Mastodon username from rOpenSci author pages</i>
----------	--

Description

Find Mastodon username from rOpenSci author pages

Usage

```
ro_masto(name)
```

Arguments

name Character. Full name as on RO author pages

Value

Character URL to mastodon profile if it exists, NA otherwise.

Examples

```
ro_masto("Steffi LaZerte")
```

slack_channels	<i>Title</i>
----------------	--------------

Description

Title

Usage

```
slack_channels(types = c("public_channel", "private_channel"))
```

Arguments

channel

Examples

```
slack_channels()
slack_channels("general")
chn <- slack_channels(types = "private_channel")
```

slack_cleanup	<i>Clean up old scheduled messages</i>
---------------	--

Description

Removes previously scheduled messages from #admin-scheduled if after the posting date.

Usage

```
slack_cleanup()
```

Examples

```
slack_posts_write("testing cleanup",
                  when = Sys.time() + lubridate::seconds(600),
                  tz = Sys.timezone())
slack_scheduled_list()

slack_cleanup()
```

slack_messages	<i>Get the last 100 messages from a channel</i>
----------------	---

Description

Get the last 100 messages from a channel

Usage

```
slack_messages(channel = NULL, channel_id = NULL)
```

Examples

```
slack_messages(channel_id = "C026GCWKA") # General
```

slack_posts_write *Write Slack message*

Description

Write a Slack message for posting now or later.

Usage

```
slack_posts_write(  
  body,  
  when = "now",  
  tz = "America/Winnipeg",  
  channel = "#testing-api",  
  dry_run = FALSE  
)
```

Arguments

body	Character. Text of message to post.
when	Character or Date/time. When to post message.
dry_run	Logical. Test run?
where	Character. Channel to post message to.

Details

See <https://docs.slack.dev/messaging/formatting-message-text#special-mentions> <https://docs.slack.dev/messaging/message-text#mentioning-users>

Value

Success message

References

- <https://docs.slack.dev/messaging/sending-and-scheduling-messages>
- <https://docs.slack.dev/messaging/sending-and-scheduling-messages#scheduling>

Examples

```
slack_posts_write("testing on Tuesday")  
slack_posts_write("testing more and more", when = "2025-05-22 14:00:00", tz = "Europe/Paris")  
slack_posts_write(  
  paste(  
    "Join us for Social Coworking and office hours next week!",  
    "",  
    ":grey_exclamation: Theme: TESTING",  
    ":hourglass_flowing_sand: When: TODAY!",
```

```

":cookie: Hosted by: USER! and cohost HOST",
"",
"You can use this time for...",
"- General coworking", sep = "\n"), when = "now")

```

slack_scheduled_list *Title*

Description

Title

Usage

```
slack_scheduled_list()
```

Examples

```
slack_scheduled_list()
```

slack_scheduled_rm *Delete a scheduled message*

Description

Delete a scheduled message

Usage

```
slack_scheduled_rm(msg = NULL, channel = NULL, id = NULL)
```

Arguments

<code>msg</code>	Data frame. Output of <code>slack_list_scheduled()</code> containing messages to remove. Should contain columns "channel" and "id"
<code>channel</code>	Character. If no msg, the Channel of the message to be deleted.
<code>id</code>	Character. If no msg, the ID of the message to be deleted.

Examples

```
# Schedule message
slack_posts_write("Testing delete msg", when = (Sys.Date() + lubridate::days(2)))

# Confirm scheduled
slack_scheduled_list()

# Remove message
slack_scheduled_list() |>
  dplyr::filter(text == "Testing delete msg") |>
  slack_scheduled_rm()

# Confirm that removed
slack_scheduled_list()

## Not run:
slack_scheduled_rm(channel = "#testing-api", id = "Q08U4S3J6QG")

## End(Not run)
```

slack_user

Fetch details on a specific users

Description

Fetch details on a specific users

Usage

```
slack_user(name, users = NULL)
```

Arguments

name Character. String to match real name to

users Data frame. Data frame of users from `slack_users()`.

Value

Data frame

Examples

```
slack_user("Steffi")
```

slack_users	<i>Fetch a list of Slack users</i>
-------------	------------------------------------

Description

Fetch a list of Slack users

Usage

```
slack_users()
```

Examples

```
u <- slack_users()
```

socials_post_issue	<i>Create a draft issue to post to Mastodon and LinkedIn</i>
--------------------	--

Description

Formats the body and title of an issue and posts it on "rosadmin/scheduled_socials". The issue will be opened in a browser for editing and confirmation. Note that issues will not be posted until the labels "draft" and "needs-review" have been removed.

Usage

```
socials_post_issue(  
  time,  
  tz = "America/Winnipeg",  
  title,  
  body,  
  where = "mastodon",  
  avoid_dups = TRUE,  
  add_hash = TRUE,  
  dry_run = FALSE,  
  open_browser = TRUE,  
  over_char_limit = stop,  
  verbose = FALSE  
)
```

Arguments

<code>time</code>	Date/time. Date and time at which the post should be made
<code>tz</code>	Character. Timezone (from <code>OlsonNames()</code>) in which to post
<code>title</code>	Character. Title of the post (<code>[Post]</code> and the date will be prepended and appended)
<code>body</code>	Character. Text to be posted (omit the YAML for posting info; <code>#RStats</code> and <code>@rstats@a.gup.pe</code> will be appended for Mastodon, <code>#RStats</code> for LinkedIn), <i>or</i> link to text file with both Mastodon and LinkedIn body text, headed by <code>— Mastodon —</code> and <code>— LinkedIn —</code> .
<code>where</code>	Character vector. Either <code>mastodon</code> and/or <code>linkedin</code> to specify which platforms this should be posted on.
<code>avoid_dups</code>	Logical. Don't post an issue if any open issue has the same title.
<code>add_hash</code>	Logical. Whether to automatically add the RStats hashtags.
<code>dry_run</code>	Logical. Whether to perform a dry run (do not post, but display draft if <code>verbose = TRUE</code>).
<code>open_browser</code>	Logical. Whether to open the issue in the browser.
<code>over_char_limit</code>	Function. Stop or warn if over the character limit?
<code>verbose</code>	Logical. If dry run, displace draft?

<code>yaml_extract</code>	<i>Extract YAML keys from block</i>
---------------------------	-------------------------------------

Description

Extract YAML keys from block

Usage

```
yaml_extract(yaml, trim = "~~~")
```

Arguments

<code>yaml</code>	Character. String from which to extract YAML keys
<code>trim</code>	Character. Text to remove from the YAML block before processing. Usually the text that defines the block.

Value

data frame of yaml keys

Examples

```
yaml_extract("~~~start: 2023-11-12\nauthor: Steffi\n~~~")
```

Index

all_users, 3

cw_checkin, 3
cw_details, 4
cw_event, 4
cw_issue, 5
cw_slack_hour, 5
cw_socials, 6

discourse_user, 7

forum_mention, 7
forum_post, 8
forum_resource, 8

gh_cache, 9
gh_issue_fetch, 9
gh_issue_fmt, 10
gh_issue_labels, 11
gh_masto, 12
gh_name, 12
gh_user, 13

li_auth, 13
li_posts_read, 14
li_posts_write, 15
li_urn_me, 16

masto2user, 16
masto_user, 17

next_date, 17

pkg_authors, 19
pkgs, 18

replace_emoji, 19
ro_masto, 20

slack_channels, 20
slack_cleanup, 21
slack_messages, 21
slack_posts_write, 22
slack_scheduled_list, 23
slack_scheduled_rm, 23
slack_user, 24
slack_users, 25
socials_post_issue, 25

yaml_extract, 26