Package ‘meetupr’

December 22, 2025

Type Package

Title Access Meetup Data

Version 0.3.1

Description Provides programmatic access to the 'Meetup' 'GraphQL' API (<https://www.meetup.com/graphql/>), enabling users to retrieve information about groups, events, and members from 'Meetup' (<https://www.meetup.com/>). Supports authentication via 'OAuth2' and includes functions for common queries and data manipulation tasks.

License MIT + file LICENSE

Depends R (>= 4.2)

Imports countrycode, cli, clipr, dplyr, fs, glue, httr2, jsonlite, lifecycle, purrr, rlang, rlist, S7, tools, stats, utils, withr, rstudioapi

Suggests covr, cyphr, ggplot2, ggwordcloud, httpuv, jose, knitr, openssl, rmarkdown, sodium, testthat, tidyr, tidytext, vcr

VignetteBuilder knitr

Config/testthat/edition 3

Config/testthat/parallel true

Encoding UTF-8

RoxygenNote 7.3.3

URL https://rladies.org/meetupr/

BugReports https://github.com/rladies/meetupr/issues

NeedsCompilation no

Author Athanasia Mo Mowinckel [aut, cre] (ORCID: <https://orcid.org/0000-0002-5756-0223>),
Erin LeDell [aut],
Olga Mierzwa-Sulima [aut],
Lucy D'Agostino McGowan [aut],
Claudia Vitolo [aut],
Gabriela De Queiroz [ctb],

Michael Beigelmacher [ctb],
Augustina Ragwitz [ctb],
Greg Sutcliffe [ctb],
Rick Pack [ctb],
Ben Ubah [ctb],
Maëlle Salmon [ctb] (ORCID: <https://orcid.org/0000-0002-2815-0399>),
Barret Schloerke [ctb] (ORCID: <https://orcid.org/0000-0001-9986-114X>),
R-Ladies Global [cph]

Maintainer Athanasia Mo Mowinckel <a.m.mowinckel@psykologi.uio.no>

Repository CRAN

Date/Publication 2025-12-22 19:50:02 UTC

Contents

find_groups *Find groups using text-based search*

Description

Search for groups on Meetup using a text query. This function allows you to find groups that match your search criteria.

Usage

```
find_groups(
  query,
  topic_id = NULL,
  category_id = NULL,
  max_results = 200,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)
```

Arguments

`query`	Character string to search for groups
`topic_id`	Numeric ID of a topic to filter groups by
`category_id`	Numeric ID of a category to filter groups by
`max_results`	Maximum number of results to return. If set to NULL, will return all available results (may take a long time).
`handle_multiples`	Character. How to handle multiple matches. One of "list" or "first", or "error". If "list", return a list-column with all matches. If "first", return only the first match.
`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	other named variables for graphql query. options are: • lat: Float • lon: Float • radius: Float • categoryId: ID • topicCategoryId: ID

Value

A tibble with group information

Examples

```
groups <- find_groups("R-Ladies")
groups
```

`find_topics` *Find topics on Meetup*

Description

Search for topics on Meetup using a query string. This function allows you to find topics that match your search criteria.

Usage

```
find_topics(
  query,
  max_results = 200,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)
```

Arguments

`query`	A string query to search for topics.
`max_results`	Maximum number of results to return. If set to NULL, will return all available results (may take a long time).
`handle_multiples`	Character. How to handle multiple matches. One of "list" or "first", or "error". If "list", return a list-column with all matches. If "first", return only the first match.
`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	Used for parameter expansion, must be empty.

Value

A data frame of topics matching the search query.

Examples

```
find_topics("R", max_results = 10)
find_topics("Data Science", max_results = 5)
```

`get_client_name` *Get Meetup OAuth Client Name*

Description

Retrieves the Meetup OAuth client name from the `MEETUPR_CLIENT_NAME` environment variable or defaults to "meetupr".

Usage

```
get_client_name()
```

Value

A string representing the client name.

`get_event` *Get information for a specified event*

Description

Get information for a specified event

Usage

```
get_event(id, extra_graphql = NULL, asis = FALSE, ...)
```

Arguments

`id`	Required event ID
`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	Should be empty. Used for parameter expansion

Value

A meetupr_event object with information about the specified event

Examples

```
event <- get_event(id = "103349942")
```

get_event_comments *Get the comments for a specified event*

Description

Get the comments for a specified event

Usage

```
get_event_comments(id, ..., extra_graphql = NULL)
```

Arguments

`id`	Required event ID
`...`	Should be empty. Used for parameter expansion
`extra_graphql`	A graphql object. Extra objects to return

Value

A tibble

Examples

```
## Not run:
comments <- get_event_comments(id = "103349942")

## End(Not run)
```

get_event_rsvps *Get the RSVPs for a specified event*

Description

Get the RSVPs for a specified event

Usage

```
get_event_rsvps(
  id,
  max_results = NULL,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)
```

Arguments

`id`	Required event ID
`max_results`	Maximum number of results to return. If set to NULL, will return all available results (may take a long time).
`handle_multiples`	Character. How to handle multiple matches. One of "list" or "first", or "error". If "list", return a list-column with all matches. If "first", return only the first match.
`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	Should be empty. Used for parameter expansion

Value

A tibble with the RSVPs for the specified event

Examples

```
rsvps <- get_event_rsvps(id = "103349942")
```

`get_group` *Get detailed information about a Meetup group*

Description

Get detailed information about a Meetup group

Usage

```
get_group(urlname, asis = FALSE)
```

Arguments

`urlname`	The URL name of the Meetup group (e.g., "rladies-lagos")
`asis`	Return the raw API response as-is without processing

Value

A list containing detailed information about the Meetup group

Examples

```
get_group("rladies-lagos")
```

get_group_events *Get the events from a meetup group*

Description

Get the events from a meetup group

Usage

```
get_group_events(
  urlname,
  status = NULL,
  date_before = NULL,
  date_after = NULL,
  max_results = NULL,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)
```

Arguments

`urlname`	Character. The name of the group as indicated in the `https://www.meetup.com/` url.
`status`	Character vector of event statuses to retrieve.
`date_before`	Datetime string in "YYYY-MM-DDTHH:MM:SSZ" (ISO8601) format. Events occurring before this date/time will be returned.
`date_after`	Datetime string in "YYYY-MM-DDTHH:MM:SSZ" (ISO8601) format. Events occurring after this date/time will be returned.
`max_results`	Maximum number of results to return. If set to NULL, will return all available results (may take a long time).
`handle_multiples`	Character. How to handle multiple matches. One of "list" or "first", or "error". If "list", return a list-column with all matches. If "first", return only the first match.
`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	Should be empty. Used for parameter expansion

Value

A tibble with the events for the specified group

Examples

```
get_group_events("rladies-lagos", "past")
get_group_events(
  "rladies-lagos",
  status = "past",
  date_before = "2023-01-01T12:00:00Z"
)
```

`get_group_members` *Get the members from a meetup group*

Description

Get the members from a meetup group

Usage

```
get_group_members(
  urlname,
  max_results = NULL,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)
```

Arguments

`urlname`	Character. The name of the group as indicated in the `https://www.meetup.com/` url.
`max_results`	Maximum number of results to return. If set to NULL, will return all available results (may take a long time).
`handle_multiples`	Character. How to handle multiple matches. One of "list" or "first", or "error". If "list", return a list-column with all matches. If "first", return only the first match.
`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	Should be empty. Used for parameter expansion

Value

A tibble with group members

Examples

```
get_group_members("rladies-lagos")
```

`get_pro` *Retrieve information about Meetup Pro networks, including groups and events.*

Description

Meetup Pro is a premium service for organizations managing multiple Meetup groups. This functionality allows you to access details about the groups within a Pro network and the events they host.

Usage

```
get_pro_groups(
  urlname,
  max_results = NULL,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)

get_pro_events(
  urlname,
  status = NULL,
  date_before = NULL,
  date_after = NULL,
  max_results = NULL,
  handle_multiples = "list",
  extra_graphql = NULL,
  asis = FALSE,
  ...
)
```

Arguments

`urlname`	Character. The name of the group as indicated in the `https://www.meetup.com/` url.
`max_results`	Maximum number of results to return. If set to NULL, will return all available results (may take a long time).
`handle_multiples`	Character. How to handle multiple matches. One of "list" or "first", or "error". If "list", return a list-column with all matches. If "first", return only the first match.

`extra_graphql`	A graphql object. Extra objects to return
`asis`	Return the raw API response as-is without processing
`...`	Should be empty. Used for parameter expansion
`status`	Which status the events should have.
`date_before`	Datetime string in "YYYY-MM-DDTHH:MM:SSZ" (ISO8601) format. Events occurring before this date/time will be returned.
`date_after`	Datetime string in "YYYY-MM-DDTHH:MM:SSZ" (ISO8601) format. Events occurring after this date/time will be returned.

Value

tibble with pro network information

A tibble with meetup pro information

Functions

- `get_pro_groups()`: retrieve groups in a pro network
- `get_pro_events()`: retrieve events from a pro network

Examples

```
urlname <- "rladies"
members <- get_pro_groups(urlname)

upcoming_events <- get_pro_events(urlname, "upcoming", max_results = 5)
```

`get_self` *Get information about the authenticated user*

Description

Retrieves detailed information about the currently authenticated Meetup user, including basic profile data, account type, subscription status, and API access permissions.

Usage

```
get_self(asis = FALSE)
```

Arguments

`asis`	Return the raw API response as-is without processing

Value

A list containing user information

Examples

```
user <- get_self()
cat("Hello", user$name, "!")
```

`local_meetupr_debug` *Temporarily enable debug mode*

Description

Sets the level of verbosity for `httr2::local_verbosity()`, and the MEETUPR_DEBUG environment variable. Can help debug issues with API requests.

Usage

```
local_meetupr_debug(verbosity = 0, env = rlang::caller_env())
```

Arguments

`verbosity`	How much debug information to show. 0 = off, 1 = basic, 2 = verbose, 3 = very verbose
`env`	The environment in which to set the variable.

Value

The old debug value (invisibly)

Examples

```
## Not run:
local_meetupr_debug(2)

# Turn off debug mode
local_meetupr_debug(0)

## End(Not run)
```

`meetupr_auth` *Authenticate and return the current user*

Description

Attempts to retrieve information about the currently authenticated user. On success a message is emitted and the user object is returned. On failure a message is shown and `NULL` is returned.

Usage

```
meetupr_auth(client_name = get_client_name())
```

Arguments

`client_name`	OAuth client name

Value

A user list (invisibly) or `NULL` on failure

`meetupr_auth_status` *Check all authentication methods and return details*

Description

Check all authentication methods and return details

Logical checker

Usage

```
meetupr_auth_status(client_name = get_client_name(), silent = FALSE)

has_auth(client_name = get_client_name())
```

Arguments

`client_name`	A string representing the name of the client. By default, it is set to `"meetupr"` and retrieved from the `MEETUPR_CLIENT_NAME` environment variable.
`silent`	logical, suppress messages if TRUE

Value

list with logicals and details for each method

Functions

- `has_auth()`: Check if any authentication method is available

Examples

```
## Not run:
# Check detailed authentication status with default client name
status_details <- meetupr_auth_status()

# Check detailed authentication status with a specific client name
status_details <- meetupr_auth_status(client_name = "custom_client")

# Suppress output messages
status_details <- meetupr_auth_status(silent = TRUE)

## End(Not run)
```

`meetupr_client` *Create a Meetup OAuth client*

Description

This function creates an httr2 OAuth client for the Meetup API. It uses environment variables or built-in credentials as fallback.

Usage

```
meetupr_client(
  client_key = NULL,
  client_secret = NULL,
  client_name = get_client_name(),
  ...
)
```

Arguments

`client_key`	Optional. The OAuth client ID.
`client_secret`	Optional. The OAuth client secret.
`client_name`	A string representing the name of the client. By default, it is set to `"meetupr"` and retrieved from the `MEETUPR_CLIENT_NAME` environment variable.
`...`	Additional arguments passed to `httr2::oauth_client()`.

Value

An httr2 OAuth client object.

`meetupr_credentials` *Manage API Keys via Environment Variables*

Description

Store and retrieve credentials using environment variables.

Usage

```
meetupr_key_set(key, value, client_name = get_client_name())

meetupr_key_get(key, client_name = get_client_name(), error = TRUE)

meetupr_key_delete(key, client_name = get_client_name())
```

Arguments

`key`	Key name: `"client_key"`, `"client_secret"`, `"encrypt_path"`, `"encrypt_pwd"`, `"jwt_token"` or `"jwt_issuer"`.
`value`	Value to be stored.
`client_name`	A string representing the name of the client. By default, it is set to `"meetupr"` and retrieved from the `MEETUPR_CLIENT_NAME` environment variable.
`error`	Throw error if key not found. Default TRUE.

Details

This is an alias of `meetupr_credentials` with hyphenated envvar names. Credentials are stored as environment variables with the pattern `{client_name}-{key}` (e.g., `meetupr-client_key`).

- `client_key`: OAuth client ID
- `client_secret`: OAuth client secret
- `encrypt_path`: Path to encrypted token file
- `encrypt_pwd`: Password for encrypted token
- `jwt_token`: JWT token for service account authentication
- `jwt_issuer`: JWT issuer, Meetup account number

Value

`meetupr_key_set()` and `meetupr_key_delete()` return NULL (invisibly), and `meetupr_key_get()` returns a character string or NA.

Functions

- `meetupr_key_set()`: Store a key in environment variables
- `meetupr_key_get()`: Retrieve a key from environment variables
- `meetupr_key_delete()`: Delete a key from environment variables

`meetupr_deauth` *Deauthorize and remove cached authentication*

Description

Remove cached OAuth tokens and optionally clear stored credentials.

Usage

```
meetupr_deauth(client_name = get_client_name())
```

Arguments

`client_name`	OAuth client name

Value

Invisible `NULL`

`meetupr_encrypt` *CI Authentication with Encrypted Token Rotation*

Description

Manage Meetup API authentication in CI environments using encrypted tokens that auto-refresh when expired.

Usage

```
meetupr_encrypt_setup(
  path = ".meetupr.rds",
  password = NULL,
  client_name = get_client_name(),
  ...
)

meetupr_encrypt_load(
  path = get_encrypted_path(),
  client_name = get_client_name(),
  password = meetupr_key_get("encrypt_pwd", client_name = client_name),
  ...
)

get_encrypted_path(client_name = get_client_name())
```

Arguments

`path`	Path to encrypted token file. Default `".meetupr.rds"`.
`password`	Encryption password. If NULL, generates random password.
`client_name`	A string representing the name of the client. By default, it is set to `"meetupr"` and retrieved from the `MEETUPR_CLIENT_NAME` environment variable.
`...`	Additional arguments to `meetupr_client()`.

Details

Setup: Run `meetupr_auth()`, then `meetupr_encrypt_setup()`. Commit the encrypted file and add password to CI secrets as `meetupr_encrypt_pwd`.

`meetupr_encrypt_load()` checks token expiration and refreshes only when needed, saving the rotated token back to the encrypted file.

Value

- `meetupr_encrypt_setup()`: Encryption password (invisibly)
- `meetupr_encrypt_load()`: httr2_token object

Functions

- `meetupr_encrypt_setup()`: Setup encrypted token for CI
- `meetupr_encrypt_load()`: Load and refresh encrypted token
- `get_encrypted_path()`: Get encrypted token path

Examples

```
## Not run:
meetupr_auth()
password <- meetupr_encrypt_setup()

# In CI
token <- meetupr_encrypt_load()

## End(Not run)
```

`meetupr_query` *Execute GraphQL query*

Description

This function executes a GraphQL query with the provided variables. It validates the variables, constructs the request, and handles any errors returned by the GraphQL API.

Usage

```
meetupr_query(graphql, ..., .envir = parent.frame())
```

Arguments

`graphql`	GraphQL query string
`...`	Variables to pass to query
`.envir`	Environment for error handling

Value

The response from the GraphQL API as a list.

Examples

```
## Not run:
query <- "
query GetUser($id: ID!) {
 user(id: $id) {
  id
 name
}
}"
meetupr_query(graphql = query, id = "12345")

## End(Not run)
```

`meetupr_req` *Create and Configure a Meetup API Request*

Description

This function prepares and configures an HTTP request for interacting with the Meetup API. It handles both interactive and non-interactive OAuth flows. In interactive mode, it uses OAuth authorization code flow. In non-interactive mode (CI/CD), it loads cached tokens.

Usage

```
meetupr_req(rate_limit = 500/60, cache = TRUE, ...)
```

Arguments

`rate_limit`	A numeric value specifying the maximum number of requests per second. Defaults to `500 / 60` (500 requests per 60 seconds).
`cache`	A logical value indicating whether to cache the OAuth token on disk. Defaults to `TRUE`.
`...`	Additional arguments passed to `meetupr_client()` for setting up the OAuth client.

Details

This function constructs an HTTP POST request directed to the Meetup API and applies appropriate OAuth authentication. The function automatically detects whether it's running in an interactive or non-interactive context:

- **Interactive**: Uses OAuth authorization code flow with browser redirect
- **Non-interactive**: Loads pre-cached token from CI environment or disk

Value

A `httr2` request object pre-configured to interact with the Meetup API.

Examples

```
## Not run:
req <- meetupr_req(cache = TRUE)

req <- meetupr_req(
  cache = FALSE,
  client_key = "your_client_key",
  client_secret = "your_client_secret"
)

## End(Not run)
```

`meetupr_schema` *Introspect the Meetup GraphQL API schema*

Description

This function performs an introspection query on the Meetup GraphQL API to retrieve the full schema details, including available query types, mutation types, and type definitions.

Usage

```
meetupr_schema(asis = FALSE)
```

Arguments

`asis`	Logical; if TRUE, returns the raw response from the API as JSON. If FALSE (default), returns the parsed schema object.

Value

If `asis` is FALSE (default), the parsed schema object with nested lists containing query types, mutation types, and type definitions. If `asis` is TRUE, a JSON string representation of the schema.

Examples

```
# Get the full schema
schema <- meetupr_schema()

# Explore what's available
names(schema)

# Get as JSON for external tools
schema_json <- meetupr_schema(asis = TRUE)
```

---

meetupr_schema_mutations

Explore available mutations in the Meetup GraphQL API

Description

This function retrieves the mutation operations available in the Meetup GraphQL API. Mutations are operations that modify data on the server (create, update, delete).

Usage

```
meetupr_schema_mutations(schema = meetupr_schema())
```

Arguments

`schema`	The schema object obtained from `meetupr_schema()`.

Value

A tibble with details about each mutation, including:

field_name Name of the mutation

description Human-readable description

args_count Number of arguments the mutation accepts

return_type The GraphQL type returned after mutation

If no mutations are available, returns a tibble with a message.

Examples

```
## Not run:
# List all available mutations
mutations <- meetupr_schema_mutations()

# Check if mutations are supported
if (nrow(mutations) > 0 && !"message" %in% names(mutations)) {
  print(mutations$field_name)
}

## End(Not run)
```

---

meetupr_schema_queries

Explore available query fields in the Meetup GraphQL API

Description

This function retrieves the root-level query fields available in the Meetup GraphQL API. These are the entry points for data fetching (e.g., `groupByUrlname`, `event`, etc.).

Usage

```
meetupr_schema_queries(schema = meetupr_schema())
```

Arguments

`schema`	The schema object obtained from `meetupr_schema()`.

Value

A tibble with details about each query field, including:

field_name Name of the query field

description Human-readable description of the field

args_count Number of arguments the field accepts

return_type The GraphQL type returned by this field

Examples

```
## Not run:
# List all available queries
queries <- meetupr_schema_queries()

# Find group-related queries
queries |>
  dplyr::filter(grepl("group", field_name, ignore.case = TRUE))

## End(Not run)
```

meetupr_schema_search *Search for types in the Meetup GraphQL API schema*

Description

This function searches across all types in the schema by name or description. Useful for discovering what data structures are available (e.g., Event, Group, Venue, Member).

Usage

```
meetupr_schema_search(pattern, schema = meetupr_schema())
```

Arguments

`pattern`	A string pattern to search for in type names and descriptions. The search is case-insensitive.
`schema`	The schema object obtained from `meetupr_schema()`.

Value

A tibble with details about matching types:

type_name Name of the type

kind GraphQL kind (OBJECT, ENUM, INTERFACE, etc.)

description Human-readable description

field_count Number of fields in the type

Examples

```
## Not run:
# Find all event-related types
meetupr_schema_search("event")

# Find location-related types
meetupr_schema_search("location")

## End(Not run)
```

`meetupr_schema_type` *Get fields for a specific type in the Meetup GraphQL API schema*

Description

This function retrieves detailed information about all fields available on a specific GraphQL type. Use this to discover what data you can query from types like Event, Group, or Member.

Usage

```
meetupr_schema_type(type_name, schema = meetupr_schema(), ...)
```

Arguments

`type_name`	The name of the type for which to retrieve fields (e.g., "Event", "Group", "Member").
`schema`	The schema object obtained from `meetupr_schema()`.
`...`	Additional arguments passed to `grepl()` for type name matching (e.g., `ignore.case = TRUE`).

Value

A tibble with details about the fields:

field_name Name of the field

description Human-readable description

type GraphQL type of the field

deprecated Logical indicating if field is deprecated

If the type is not found, throws an error. If multiple types match, returns a tibble of matching type names. If the type has no fields, returns a message.

Examples

```
## Not run:
# Get all fields on the Event type
event_fields <- meetupr_schema_type("Event")

# Find deprecated fields
event_fields |>
  dplyr::filter(deprecated)

# Pass cached schema to avoid repeated introspection
schema <- meetupr_schema()
group_fields <- meetupr_schema_type("Group", schema = schema)
venue_fields <- meetupr_schema_type("Venue", schema = schema)

## End(Not run)
```

meetupr_sitrep *Show meetupr authentication status*

Description

This function checks the authentication status for the Meetup API. It provides feedback on whether credentials are configured correctly, tests API connectivity, and shows available authentication methods.

Usage

```
meetupr_sitrep()
```

Value

Invisibly returns a list with authentication status details.

Examples

```
meetupr_sitrep()
```

use_gha *GitHub Actions workflow helpers*

Description

Create GitHub Actions workflows for JWT and encrypted-token CI auth.

Usage

```
use_gha_jwt_token(
  client_name = get_client_name(),
  jwt = "MEETUPR_JWT_TOKEN",
  client_key = "MEETUPR_client_key",
  overwrite = FALSE,
  filename = ".github/workflows/meetupr-jwt-token.yml"
)

use_gha_encrypted_token(
  token_path = get_encrypted_path(),
  overwrite = FALSE,
  filename = ".github/workflows/meetupr-rotate-token.yml"
)
```

Arguments

`client_name`	A string representing the name of the client. By default, it is set to `"meetupr"` and retrieved from the `MEETUPR_CLIENT_NAME` environment variable.
`jwt`	Name of the GitHub secret that contains the JWT token. Defaults to "MEETUPR_JWT_TOKEN".
`client_key`	Name of the GitHub secret that contains the client id. Defaults to "MEETUPR_client_key".
`overwrite`	If FALSE and file exists, function aborts.
`filename`	Path to write the workflow YAML file. Defaults to appropriate path in `.github/workflows/`.
`token_path`	Path to the encrypted token file in the repo. Defaults to get_encrypted_path().

Details

These helpers write workflow YAML into `.github/workflows/` so you can enable non-interactive authentication for meetupr in GitHub Actions.

Which helper to use:

- use_gha_jwt_token(): For Meetup Pro accounts that use a static JWT token. The generated workflow provides a JWT and client id as repository secrets so the runner can authenticate via JWT.
- use_gha_encrypted_token(): For regular OAuth apps. This creates a rotation workflow that loads an encrypted `.meetupr.rds` token, refreshes it when needed, and writes the rotated encrypted file back to the repo. Requires a decryption password stored as a repository secret (default: MEETUPR_ENCRYPT_PWD).

Important notes:

- These functions only write the workflow YAML. They do not create repository secrets or commit/push files for you.
- Review the generated YAML and add the required secrets in your repository settings before enabling the workflow.

Value

Invisibly returns the path to the created workflow file.

Functions

- `use_gha_jwt_token()`: Create workflow for JWT token auth.
- `use_gha_encrypted_token()`: Create workflow for encrypted token rotation.

Index