Package: googlesecretmanager (via r-universe)

May 12, 2026

Title Manage Google Cloud Platform Secrets with R

Version 0.1.0

Description An R package for interacting with Google Cloud Secret Manager, providing a secure way to manage and access secrets in your R applications.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Imports cli, dplyr, gargle (>= 1.5.2), glue, httr (>= 1.4.0), jsonlite (>= 1.8.0), lifecycle, purrr, rlang, utils

Depends R (>= 4.0)

URL https://brancengregory.github.io/googlesecretmanager/

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

Date/Publication 2025-06-08 00:08:15 UTC

RemoteUrl https://github.com/brancengregory/googlesecretmanager

RemoteRef HEAD

RemoteSha 5b0a05127e77344ae4b245d1a95b2b5ebfd6cd89

Contents

`as_secret` *As Secret*

Description

As Secret

Usage

```
as_secret(x, ...)
```

Arguments

`x`	A string
`...`	Unused argument needed for method

`as_secret.character` *As Secret Character*

Description

As Secret Character

Usage

```
## S3 method for class 'character'
as_secret(x, ...)
```

Arguments

x	A string
...	Unused argument needed for method

---

print.sm_secret *Print sm_secret*

Description

Print sm_secret

Usage

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

Arguments

x	A sm_secret object
...	Additional arguments passed to method

---

print.sm_tbl *Print sm_tbl*

Description

Print sm_tbl

Usage

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

Arguments

x	A sm_tbl object
...	Additional arguments passed to method

sm_api_key *Retrieve the configured API key*

Description

Returns the currently configured API key for the `googlesecretmanager` package. Note: API keys have limited use with Secret Manager, which primarily relies on OAuth2.0.

Usage

```
sm_api_key()
```

Value

A string containing the API key, or `NULL` if no key is configured.

Examples

```
## Not run:
# Configure API key first (if applicable)
# sm_auth_configure(api_key = "YOUR_API_KEY")
key <- sm_api_key()
if (!is.null(key)) {
  print(key)
}

## End(Not run)
```

sm_auth *Authenticate with Google Secret Manager*

Description

This function handles authentication with Google Cloud Secret Manager. It's called automatically when the first token is needed, or it can be called directly by the user to pre-authenticate or to switch identities, scopes, or authentication methods (e.g., user OAuth, service account token).

Following `gargle` best practices for sensitive APIs, `googlesecretmanager` does **not** come with a built-in OAuth client or API key. You must configure your own via `sm_auth_configure()` or provide a service account token via the `path` argument.

Usage

```
sm_auth(
  email = gargle::gargle_oauth_email(),
  path = NULL,
  scopes = "https://www.googleapis.com/auth/secretmanager",
  cache = gargle::gargle_oauth_cache(),
  use_oob = gargle::gargle_oob_default(),
  token = NULL
)
```

Arguments

email	Optional. The email address of the Google identity you want to authenticate with. Useful for selecting a specific account if you have multiple, or for non-interactive authentication. If NULL, gargle will try to obtain it from the "gargle_oauth_email" option or allow you to choose from a list in interactive sessions. See gargle::gargle_oauth_email() for more details.
path	Optional. Path to a service account token (JSON file) or a pre-existing token. If provided, this will be used for authentication instead of the OAuth flow. See gargle::token_fetch() for details on accepted formats.
scopes	The OAuth scopes to request. For Secret Manager, a common scope is "https://www.googleapis.com/ or the broader "https://www.googleapis.com/auth/cloud-platform". Defaults to "https://www.googleapis.com/auth/secretmanager".
cache	The location of the OAuth token cache. Defaults to gargle::gargle_oauth_cache().
use_oob	Whether to prefer "out-of-band" (OOB) authentication. Defaults to gargle::gargle_oob_default(). Useful for non-interactive sessions where a browser cannot be easily launched.
token	A pre-existing token object (e.g., from httr::Token2.0 or another gargle-using package). If provided, this token will be used directly.

Value

Invisibly returns NULL. The main effect is to configure authentication state for the package.

Examples

```
## Not run:
# To configure your own OAuth client (do this once per project/user):
sm_auth_configure(
 path = "/path/to/your/oauth-client-secret.json"
)

# Authenticate (often not needed explicitly, called by API functions):
sm_auth()

# Authenticate with a specific user:
sm_auth(email = "my_user@example.com")

# Authenticate using a service account:
```

```
sm_auth(path = "/path/to/your/service-account-key.json")

# Authenticate using a pre-fetched token (like googleCloudStorageR example):
token <- gargle::token_fetch(
  scopes = "https://www.googleapis.com/auth/cloud-platform"
)
sm_auth(token = token)

## End(Not run)
```

`sm_auth_configure` *Configure OAuth client and API key for Secret Manager*

Description

This function allows advanced users to provide their own OAuth client ID and secret (from a JSON file downloaded from Google Cloud Console) or an API key.

OAuth Client: For `googlesecretmanager`, providing your own OAuth client is **highly recommended** as the package does not ship with a default client due to the sensitive nature of the API.

API Key: While the `gargle` framework supports API keys, most Secret Manager operations require OAuth2.0 authentication. An API key might be useful for very limited, typically read-only, public data scenarios, which are rare for Secret Manager. It's included for structural consistency with `gargle` but may have limited direct use for this package.

Usage

```
sm_auth_configure(path = NULL, client = NULL, api_key = NULL, app = NULL)
```

Arguments

`path`	Path to a JSON file containing the OAuth client ID and secret. This is the recommended way to configure an OAuth client.
`client`	An `httr::oauth_app` object or `gargle::gargle_oauth_client` object. Alternatively, provide `path`.
`api_key`	A string representing your Google Cloud API key.
`app`	Deprecated. Use `client` instead.

Value

Invisibly returns the updated auth configuration (an `AuthState` object).

Examples

```
## Not run:
# Configure with an OAuth client downloaded from GCP
sm_auth_configure(
  path = "/path/to/your/oauth-client-secret.json"
)

# To configure with an API key (less common for Secret Manager):
# sm_auth_configure(api_key = "YOUR_API_KEY")

# Check configured client:
sm_oauth_client()

## End(Not run)
```

`sm_deauth` *De-authenticate from Secret Manager*

Description

Clears the current Secret Manager token. This means the next API request that requires authentication will trigger the authentication process anew (e.g., by calling `sm_auth()`).

Since Secret Manager generally requires authentication for all its significant operations (and doesn't typically use API keys for accessing secrets), de-authentication primarily serves to clear the current user's session or force a re-authentication.

Usage

```
sm_deauth()
```

Value

Invisibly returns `NULL`.

Examples

```
## Not run:
sm_deauth()
# Next API call will re-trigger auth
# list_secrets() # (Assuming this is a function in your package)

## End(Not run)
```

sm_endpoint *Secret Manager Endpoint*

Description

Secret Manager Endpoint

Usage

```
sm_endpoint(i)
```

Arguments

i	The index of the endpoint

sm_endpoints *List Secret Manager endpoints*

Description

The googlesecretmanager package stores a named list of Secret Manager API v1 endpoints (or "methods", using Google's vocabulary) internally and these functions expose this data.

- `sm_endpoint()` returns one endpoint, i.e. it uses `[[`.
- `sm_endpoints()` returns a list of endpoints, i.e. it uses `[`.

The names of this list (or the `id` sub-elements) are the nicknames that can be used to specify an endpoint in request_generate(). For each endpoint, we store its nickname or `id`, the associated HTTP verb, the `path`, and details about the parameters. This list is derived programmatically from the Secret Manager API v1 Discovery Document (`https://www.googleapis.com/discovery/v1/apis/secretmanager/v1/res` using the approach described in the Discovery Documents section of the gargle vignette Request helper functions.

Usage

```
sm_endpoints(i = NULL)
```

Arguments

i	The name(s) or integer index(ices) of the endpoints to return. `i` is optional for `sm_endpoints()` and, if not given, the entire list is returned.

Value

One or more of the Secret Manager API v1 endpoints that are used internally by googlesecretmanager.

Examples

```
str(head(sm_endpoints(), 3), max.level = 2)
sm_endpoint("secretmanager.projects.secrets.versions.destroy")
sm_endpoint(4)
```

`sm_oauth_client` *Retrieve the configured OAuth client*

Description

Returns the currently configured OAuth client for the `googlesecretmanager` package. This client is used in the OAuth flow to obtain tokens. By default, this will be `NULL` until configured by the user via `sm_auth_configure()`.

Usage

```
sm_oauth_client()
```

Value

A `gargle_oauth_client` object, or `NULL` if no client is configured.

Examples

```
## Not run:
# Configure client first
# sm_auth_configure(path = "/path/to/your/client.json")
client <- sm_oauth_client()
if (!is.null(client)) {
  print(client)
}

## End(Not run)
```

`sm_project_set` *Set Default Google Cloud Project ID*

Description

Sets the default Google Cloud Project ID for subsequent Secret Manager operations within the current R session. Functions will automatically use this project ID unless explicitly overridden.

Usage

```
sm_project_set(project_id)
```

Arguments

project_id	A single character string specifying the Google Cloud Project ID (e.g., "my-gcp-project-123").

Value

The project_id character string, invisibly. Called for its side-effect of setting the default project.

Examples

```
## Not run:
sm_project_set("my-production-project")
# Now, sm_secret_ls() will default to "my-production-project"

sm_project_get() # "my-production-project"

# You can always override the default for specific calls:
sm_secret_get("my-secret", project_id = "another-project")

## End(Not run)
```

---

sm_secret_create *Create a Secret*

Description

Creates a new Secret containing no SecretVersions.

Usage

```
sm_secret_create(
  secret_id,
  project_id = sm_project_get(),
  replication = list(automatic = list()),
  labels = NULL,
  ...
)
```

Arguments

secret_id	Required. A unique identifier for the secret within the project. Must be a string with a maximum length of 255 characters and can contain uppercase and lowercase letters, numerals, and the hyphen (-) and underscore (_) characters.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
replication	Required. The replication policy for the secret data. Must be a list with either automatic or user_managed configuration.
labels	Optional. Labels to attach to the secret.
...	Additional arguments for methods.

Value

An sm_secret object representing the created secret.

---

sm_secret_delete *Delete a Secret*

Description

Deletes a Secret and all of its versions.

Usage

```
sm_secret_delete(secret, project_id = sm_project_get(), ...)

## S3 method for class 'character'
sm_secret_delete(secret, project_id = sm_project_get(), ...)

## S3 method for class 'sm_secret'
sm_secret_delete(secret, project_id = sm_project_get(), ...)
```

Arguments

secret	The secret to delete. Can be a secret ID (character string) or an existing sm_secret object.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
...	Additional arguments for methods.

Value

Invisibly returns NULL.

---

sm_secret_get *Get Secret Metadata*

Description

Retrieves metadata for a specific Secret Manager secret.

Usage

```
sm_secret_get(x, project_id = sm_project_get(), ...)

## S3 method for class 'character'
sm_secret_get(x, project_id = sm_project_get(), ...)

## S3 method for class 'sm_secret'
sm_secret_get(x, project_id = sm_project_get(), ...)
```

Arguments

x	The identifier for the secret. Can be a secret ID (character string) or an existing sm_secret object to refresh its metadata.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
...	Additional arguments for methods.

---

sm_secret_ls *List Secrets in a Project*

Description

Lists metadata for all Secrets in a given Google Cloud Project.

Usage

```
sm_secret_ls(project_id = sm_project_get(), filter = NULL, ...)
```

Arguments

project_id	The Google Cloud Project ID. Defaults to sm_project_get().
filter	Optional. A filter string, adhering to Secret Manager's List-operation filtering rules.
...	Additional arguments for methods.

---

sm_secret_ls.character *List Secrets in a Project*

Description

Lists metadata for all Secrets in a given Google Cloud Project.

Usage

```
## S3 method for class 'character'
sm_secret_ls(project_id = sm_project_get(), filter = NULL, ...)
```

Arguments

project_id	The Google Cloud Project ID. Defaults to sm_project_get().
filter	Optional. A filter string, adhering to Secret Manager's List-operation filtering rules.
...	Additional arguments for methods.

`sm_secret_update` *Update a Secret*

Description

Updates metadata of an existing Secret.

Usage

```
sm_secret_update(
  secret,
  project_id = sm_project_get(),
  replication = NULL,
  labels = NULL,
  etag = NULL,
  ...
)

## S3 method for class 'character'
sm_secret_update(
  secret,
  project_id = sm_project_get(),
  replication = NULL,
  labels = NULL,
  etag = NULL,
  ...
)

## S3 method for class 'sm_secret'
sm_secret_update(
  secret,
  project_id = sm_project_get(),
  replication = NULL,
  labels = NULL,
  etag = NULL,
  ...
)
```

Arguments

`secret`	The secret to update. Can be a secret ID (character string) or an existing `sm_secret` object.
`project_id`	The Google Cloud Project ID. Defaults to `sm_project_get()`.
`replication`	Optional. The replication policy for the secret data. Must be a list with either `automatic` or `user_managed` configuration.
`labels`	Optional. Labels to attach to the secret.

etag	Optional. The etag of the secret. If provided, the update will only succeed if the secret's current etag matches this value.
...	Additional arguments for methods.

Value

An sm_secret object representing the updated secret.

---

sm_secret_version_add *Add a Secret Version*

Description

Adds a new version to an existing Secret.

Usage

```
sm_secret_version_add(secret, payload, project_id = sm_project_get(), ...)

## S3 method for class 'character'
sm_secret_version_add(secret, payload, project_id = sm_project_get(), ...)

## S3 method for class 'sm_secret'
sm_secret_version_add(secret, payload, project_id = sm_project_get(), ...)
```

Arguments

secret	The secret to add a version to. Can be a secret ID (character string) or an existing sm_secret object.
payload	The secret data to store. Will be base64 encoded.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
...	Additional arguments for methods.

Value

An sm_secret_version object representing the new version.

sm_secret_version_delete *Delete a Secret Version*

Description

Deletes a Secret Version.

Usage

```
sm_secret_version_delete(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)

## S3 method for class 'character'
sm_secret_version_delete(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)

## S3 method for class 'sm_secret'
sm_secret_version_delete(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)
```

Arguments

secret	The secret containing the version. Can be a secret ID (character string) or an existing sm_secret object.
version_id	The version ID to delete.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
...	Additional arguments for methods.

Value

Invisibly returns NULL.

`sm_secret_version_disable` *Disable a Secret Version*

Description

Disables a Secret Version.

Usage

```
sm_secret_version_disable(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)

## S3 method for class 'character'
sm_secret_version_disable(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)

## S3 method for class 'sm_secret'
sm_secret_version_disable(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)
```

Arguments

`secret`	The secret containing the version. Can be a secret ID (character string) or an existing `sm_secret` object.
`version_id`	The version ID to disable.
`project_id`	The Google Cloud Project ID. Defaults to `sm_project_get()`.
`...`	Additional arguments for methods.

Value

An `sm_secret_version` object representing the disabled version.

`sm_secret_version_enable` *Enable a Secret Version*

Description

Enables a Secret Version.

Usage

```
sm_secret_version_enable(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)

## S3 method for class 'character'
sm_secret_version_enable(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)

## S3 method for class 'sm_secret'
sm_secret_version_enable(
  secret,
  version_id,
  project_id = sm_project_get(),
  ...
)
```

Arguments

`secret`	The secret containing the version. Can be a secret ID (character string) or an existing `sm_secret` object.
`version_id`	The version ID to enable.
`project_id`	The Google Cloud Project ID. Defaults to `sm_project_get()`.
`...`	Additional arguments for methods.

Value

An `sm_secret_version` object representing the enabled version.

sm_secret_version_get *Get Secret Version Metadata*

Description

Gets metadata for a specific Secret Version.

Usage

```
sm_secret_version_get(secret, version_id, project_id = sm_project_get(), ...)

## S3 method for class 'character'
sm_secret_version_get(secret, version_id, project_id = sm_project_get(), ...)

## S3 method for class 'sm_secret'
sm_secret_version_get(secret, version_id, project_id = sm_project_get(), ...)
```

Arguments

secret	The secret containing the version. Can be a secret ID (character string) or an existing sm_secret object.
version_id	The version ID to get. Can be "latest" to get the latest version.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
...	Additional arguments for methods.

Value

An sm_secret_version object representing the version metadata.

sm_secret_version_ls *List Secret Versions*

Description

Lists metadata for all Secret Versions associated with a given Secret.

Usage

```
sm_secret_version_ls(secret, project_id = sm_project_get(), filter = NULL, ...)

## S3 method for class 'sm_secret'
sm_secret_version_ls(secret, project_id = sm_project_get(), filter = NULL, ...)

## S3 method for class 'character'
sm_secret_version_ls(secret, project_id = sm_project_get(), filter = NULL, ...)
```

Arguments

secret	The secret for which to list versions. Can be an sm_secret object or a character string representing the secret ID.
project_id	The Google Cloud Project ID. Defaults to sm_project_get().
filter	Optional. A filter string for secret versions.
...	Additional arguments for methods.

---

sm_token *Provide a token for Secret Manager API requests*

Description

Retrieves the current token for Secret Manager. If authentication is active (.sm_auth$auth_active is TRUE) and no token is cached, it will trigger sm_auth() to obtain one.

This function is typically used by other package functions that make API requests.

Usage

```
sm_token()
```

Value

An httr::config object containing the httr::Token2.0 object, or NULL if auth is inactive.

Examples

```
## Not run:
# Configure auth first if needed (e.g., with your client ID)
# sm_auth_configure(path = "path/to/client.json")
# sm_auth() # or let it be called automatically

token <- sm_token()
if (!is.null(token)) {
  # Use token in httr::GET() or other API calls
}

## End(Not run)
```

sm_user *Get information about the authenticated user*

Description

Provides information about the Google identity associated with the current token. This usually includes the email address. It attempts to retrieve this information from the token itself.

Usage

```
sm_user(token = NULL)
```

Arguments

token	An optional token object. If NULL, uses the cached token obtained by sm_token().

Value

A list containing user information (e.g., email) or NULL if no token is available or user information cannot be parsed.

Examples

```
## Not run:
# Authenticate first
# sm_auth()

user_info <- sm_user()
if (!is.null(user_info)) {
  print(user_info$email)
}

## End(Not run)
```

Index