Package 'egnyte'

May 8, 2026

Title Read and Write Files from 'Egnyte'

Version 0.1.2

Description Provides functions to read and write files from 'Egnyte' cloud storage using the 'Egnyte' API <https://developers.egnyte.com/docs>. Supports both API key and 'OAuth' 2.0 authentication for file transfer operations.

Depends R (>= 4.1.0)

License Apache License 2.0 | file LICENSE

URL https://atorus-research.github.io/egnyte/

BugReports https://github.com/atorus-research/egnyte/issues

Encoding UTF-8

RoxygenNote 7.3.2

Imports cli, httr2

Suggests haven, jsonlite, knitr, readr, readxl, rmarkdown, testthat (>= 3.0.0), withr, writexl

VignetteBuilder knitr

Config/testthat/edition 3

NeedsCompilation no

Author Mike Stackhouse [aut, cre] (ORCID: <https://orcid.org/0000-0001-6030-723X>)

Maintainer Mike Stackhouse <mike.stackhouse@atorusresearch.com>

Repository CRAN

Date/Publication 2026-02-09 19:20:13 UTC

Contents

`eg_auth` *Set Egnyte Authentication Credentials*

Description

Stores Egnyte domain and API key for use in subsequent API calls. Credentials can also be set via environment variables `EGNYTE_DOMAIN` and `EGNYTE_API_KEY`.

Usage

```
eg_auth(domain, api_key)
```

Arguments

`domain`	Your Egnyte domain (the subdomain part of yourcompany.egnyte.com)
`api_key`	Your Egnyte API key or OAuth access token

Value

Invisibly returns a list with the stored credentials.

Examples

```
## Not run:
eg_auth("mycompany", "my_api_key_here")

## End(Not run)
```

eg_list *List Files in an Egnyte Directory*

Description

Returns a character vector of full file paths within the specified Egnyte directory.

Usage

```
eg_list(path, recursive = FALSE)
```

Arguments

path	The Egnyte path to the directory (e.g., "/Shared/Documents").
recursive	If TRUE, recursively list files in subdirectories. Defaults to FALSE.

Value

A character vector of full file paths.

Examples

```
## Not run:
# List files in a directory
eg_list("/Shared/Documents")

# Recursively list all files
eg_list("/Shared/Documents", recursive = TRUE)

## End(Not run)
```

eg_oauth_app *Set Up OAuth Application Credentials*

Description

Configures the OAuth 2.0 application credentials for Egnyte authentication. These credentials are obtained by registering your application at https://developers.egnyte.com.

Usage

```
eg_oauth_app(
  domain,
  client_id,
  client_secret,
  redirect_uri = "https://localhost/callback"
)
```

Arguments

`domain`	Your Egnyte domain (the subdomain of yourcompany.egnyte.com).
`client_id`	The API key (client ID) from your registered application.
`client_secret`	The client secret from your registered application.
`redirect_uri`	The redirect URI configured for your app in the Egnyte developer portal. Must be HTTPS. Defaults to `https://localhost/callback`.

Details

After registering at https://developers.egnyte.com, you will receive a client_id and client_secret. Your application must be approved by Egnyte before it becomes active.

Important: You must configure the same redirect_uri in your Egnyte app settings. Egnyte requires all redirect URIs to be HTTPS.

During development, your API key only works with your registered Egnyte domain. After certification, it works with all Egnyte domains.

Value

Invisibly returns a list with the OAuth app configuration.

See Also

`eg_oauth_authorize()` to complete the OAuth flow.

Examples

```
## Not run:
eg_oauth_app(
  domain = "mycompany",
  client_id = "your_client_id",
  client_secret = "your_client_secret",
  redirect_uri = "https://your-registered-redirect.com/callback"
)

## End(Not run)
```

`eg_oauth_authorize` *Authorize with Egnyte via OAuth 2.0*

Description

Initiates the OAuth 2.0 Authorization Code flow. This opens a browser window for the user to log in and authorize the application, then prompts for the authorization code to exchange for access tokens.

Usage

```
eg_oauth_authorize(scope = "Egnyte.filesystem")
```

Arguments

scope	Character vector of OAuth scopes to request. Defaults to "Egnyte.filesystem" for file operations. Other scopes include "Egnyte.link", "Egnyte.user", and "Egnyte.projectfolders".

Details

The OAuth flow:

1. Opens a browser to Egnyte's authorization page
2. User logs in (via SSO if configured) and approves access
3. Egnyte redirects to your configured redirect_uri with ?code=...
4. Copy the code from the URL and paste it when prompted
5. The code is exchanged for access and refresh tokens

Note: Egnyte requires HTTPS redirect URIs. After authorization, you'll be redirected to your configured URI. Copy the code parameter from the URL (everything after code= and before any &).

Access tokens expire after 30 days. Use eg_oauth_refresh() to obtain a new token using the refresh token.

Value

Invisibly returns a list containing access_token, refresh_token, token_type, and expires_in.

See Also

eg_oauth_app() to configure the OAuth application first.

Examples

```
## Not run:
# First set up your OAuth app
eg_oauth_app("mycompany", "client_id", "client_secret",
             redirect_uri = "https://your-app.com/callback")

# Then authorize (opens browser, prompts for code)
eg_oauth_authorize()

# Now you can use eg_read() and eg_write()
eg_read("/Shared/Documents/file.txt", "local.txt")

## End(Not run)
```

`eg_oauth_password` *Authenticate with Egnyte Using Username and Password*

Description

Uses the OAuth 2.0 Resource Owner Password flow to obtain an access token directly using Egnyte credentials. This is simpler than the Authorization Code flow and doesn't require browser interaction.

Usage

```
eg_oauth_password(username = NULL, password = NULL)
```

Arguments

`username`	Egnyte username. If NULL, checks the `EGNYTE_USERNAME` environment variable.
`password`	Egnyte password. If NULL, checks the `EGNYTE_PASSWORD` environment variable.

Details

This flow is intended for internal applications where the user trusts the application with their credentials. The username and password are sent directly to Egnyte's token endpoint.

You must first configure the OAuth app with `eg_oauth_app()`.

Credentials can be provided via:

- Function arguments
- Environment variables: `EGNYTE_USERNAME`, `EGNYTE_PASSWORD`
- Interactive prompt (if running interactively and not provided)

Note: This flow does not return a refresh token. When the access token expires (after 30 days), you'll need to authenticate again.

Value

Invisibly returns a list containing `access_token` and `expires_in`.

See Also

`eg_oauth_app()` to configure the OAuth application first.

Examples

```
## Not run:
# Set up your OAuth app first
eg_oauth_app("mycompany", "client_id", "client_secret")

# Authenticate with username/password
eg_oauth_password("myuser", "mypassword")

# Or use environment variables
Sys.setenv(EGNYTE_USERNAME = "myuser", EGNYTE_PASSWORD = "mypassword")
eg_oauth_password()

# Now you can use eg_read() and eg_write()
eg_read("/Shared/Documents/file.txt", "local.txt")

## End(Not run)
```

`eg_oauth_refresh` *Refresh OAuth Access Token*

Description

Uses the stored refresh token to obtain a new access token without requiring user interaction.

Usage

```
eg_oauth_refresh()
```

Details

Access tokens expire after 30 days. Call this function to obtain a new access token using the refresh token that was stored during the initial authorization.

If the refresh token is also expired or revoked (e.g., user changed password), you will need to run `eg_oauth_authorize()` again.

Value

Invisibly returns a list containing the new `access_token`, `refresh_token`, `token_type`, and `expires_in`.

See Also

`eg_oauth_authorize()` for the initial authorization.

Examples

```
## Not run:
# Refresh the access token
eg_oauth_refresh()

## End(Not run)
```

`eg_read` *Download a File from Egnyte*

Description

Downloads a file from Egnyte cloud storage to a local path.

Usage

```
eg_read(path, destfile = NULL)
```

Arguments

`path`	The Egnyte path to the file (e.g., "/Shared/Documents/file.txt").
`destfile`	Local file path where the file will be saved. If NULL (default), the file is downloaded to a temporary file.

Value

The local file path (invisibly).

Examples

```
## Not run:
# Download to a specific location
eg_read("/Shared/Documents/report.pdf", "local_report.pdf")

# Download to a temp file
local_path <- eg_read("/Shared/Documents/data.csv")
read.csv(local_path)

## End(Not run)
```

`eg_read_file` *Read Data Files from Egnyte*

Description

These functions download data files from Egnyte and read them into R using the appropriate package. Each function is a thin wrapper that handles the file transfer, then delegates to the underlying reader.

Usage

```
eg_read_csv(path, ...)

eg_read_delim(path, delim = "\t", ...)

eg_read_sas(path, ...)

eg_read_xpt(path, ...)

eg_read_stata(path, ...)

eg_read_spss(path, ...)

eg_read_excel(path, sheet = NULL, ...)

eg_read_rds(path)
```

Arguments

`path`	The Egnyte path to the file (e.g., "/Shared/Data/myfile.csv").
`...`	Additional arguments passed to the underlying read function.
`delim`	The field delimiter. Defaults to tab (`"\t"`).
`sheet`	Sheet to read. Either a string (sheet name) or an integer (sheet position). Defaults to the first sheet.

Details

Each function requires an optional package to be installed:

Function	Package	Underlying Function
`eg_read_csv()`	readr	`readr::read_csv()`
`eg_read_delim()`	readr	`readr::read_delim()`
`eg_read_excel()`	readxl	`readxl::read_excel()`
`eg_read_sas()`	haven	`haven::read_sas()`
`eg_read_xpt()`	haven	`haven::read_xpt()`
`eg_read_stata()`	haven	`haven::read_dta()`

eg_read_spss()	haven	haven::read_sav()
eg_read_rds()	(base R)	readRDS()

All arguments passed through ... are forwarded to the underlying function, so you can use any options those functions support (e.g., col_types for readr functions, encoding for haven functions).

Value

- For tabular data: A tibble (or data frame for RDS files containing data frames).
- For RDS files: The R object stored in the file.

Haven-based readers (eg_read_sas(), eg_read_xpt(), eg_read_stata(), eg_read_spss()) return tibbles with labelled columns preserving variable labels and formats from the source file.

See Also

- eg_read() for downloading raw files without parsing
- eg_write_file for writing data files to Egnyte

Examples

```
## Not run:
# CSV files
df <- eg_read_csv("/Shared/Data/mydata.csv")
df <- eg_read_csv("/Shared/Data/mydata.csv", col_types = "ccn")

# Delimited files
df <- eg_read_delim("/Shared/Data/mydata.txt")
df <- eg_read_delim("/Shared/Data/mydata.txt", delim = "|")

# Excel files
df <- eg_read_excel("/Shared/Data/workbook.xlsx")
df <- eg_read_excel("/Shared/Data/workbook.xlsx", sheet = "Summary")

# SAS files
df <- eg_read_sas("/Shared/Data/mydata.sas7bdat")
df <- eg_read_xpt("/Shared/Data/mydata.xpt")

# Stata files
df <- eg_read_stata("/Shared/Data/mydata.dta")

# SPSS files
df <- eg_read_spss("/Shared/Data/mydata.sav")

# RDS files (any R object)
obj <- eg_read_rds("/Shared/Data/model.rds")

## End(Not run)
```

eg_write *Upload a File to Egnyte*

Description

Uploads a local file to Egnyte cloud storage.

Usage

```
eg_write(file, path, overwrite = FALSE)
```

Arguments

file	Local path to the file to upload.
path	The Egnyte destination path (e.g., "/Shared/Documents/file.txt").
overwrite	If FALSE (default), the upload will fail if a file already exists at the destination. Set to TRUE to replace existing files.

Value

The Egnyte path (invisibly).

Examples

```
## Not run:
# Upload a file
eg_write("local_report.pdf", "/Shared/Documents/report.pdf")

# Overwrite an existing file
eg_write("updated_data.csv", "/Shared/Data/data.csv", overwrite = TRUE)

## End(Not run)
```

eg_write_file *Write Data Files to Egnyte*

Description

These functions write R objects to data files and upload them to Egnyte. Each function is a thin wrapper that writes to a temporary file using the appropriate package, then handles the upload.

Usage

```
eg_write_csv(x, path, overwrite = FALSE, ...)

eg_write_delim(x, path, delim = "\t", overwrite = FALSE, ...)

eg_write_xpt(x, path, overwrite = FALSE, ...)

eg_write_stata(x, path, overwrite = FALSE, ...)

eg_write_spss(x, path, overwrite = FALSE, ...)

eg_write_excel(x, path, overwrite = FALSE, ...)

eg_write_rds(x, path, overwrite = FALSE, compress = TRUE)
```

Arguments

x	A data frame (or R object for `eg_write_rds()`) to write. For `eg_write_excel()`, can also be a named list of data frames to create multiple sheets.
path	The Egnyte destination path (e.g., "/Shared/Data/results.csv").
overwrite	If `FALSE` (default), fails if a file already exists at `path`. Set to `TRUE` to replace existing files.
...	Additional arguments passed to the underlying write function.
delim	The field delimiter. Defaults to tab (`"\t"`).
compress	Compression type for RDS files. See `saveRDS()` for options. Defaults to `TRUE`.

Details

Each function requires an optional package to be installed:

Function	Package	Underlying Function
`eg_write_csv()`	readr	`readr::write_csv()`
`eg_write_delim()`	readr	`readr::write_delim()`
`eg_write_excel()`	writexl	`writexl::write_xlsx()`
`eg_write_xpt()`	haven	`haven::write_xpt()`
`eg_write_stata()`	haven	`haven::write_dta()`
`eg_write_spss()`	haven	`haven::write_sav()`
`eg_write_rds()`	(base R)	`saveRDS()`

All arguments passed through ... are forwarded to the underlying function.

Note on SAS files: Haven can only write SAS transport files (.xpt), not native SAS data files (.sas7bdat). Transport files are compatible with SAS and can be read back with `eg_read_xpt()` or `haven::read_xpt()`.

Value

The Egnyte path (invisibly).

See Also

- eg_write() for uploading raw files without conversion
- eg_read_file for reading data files from Egnyte

Examples

```
## Not run:
# CSV files
eg_write_csv(mtcars, "/Shared/Data/mtcars.csv")
eg_write_csv(mtcars, "/Shared/Data/mtcars.csv", overwrite = TRUE)

# Delimited files
eg_write_delim(mtcars, "/Shared/Data/mtcars.tsv")
eg_write_delim(mtcars, "/Shared/Data/mtcars.txt", delim = "|")

# Excel files
eg_write_excel(mtcars, "/Shared/Data/mtcars.xlsx")

# Multiple sheets
eg_write_excel(
  list(cars = mtcars, flowers = iris),
  "/Shared/Data/workbook.xlsx"
)

# SAS transport files
eg_write_xpt(mtcars, "/Shared/Data/mtcars.xpt")

# Stata files
eg_write_stata(mtcars, "/Shared/Data/mtcars.dta")

# SPSS files
eg_write_spss(mtcars, "/Shared/Data/mtcars.sav")

# RDS files (any R object)
eg_write_rds(my_model, "/Shared/Data/model.rds")
eg_write_rds(large_data, "/Shared/Data/data.rds", compress = "xz")

## End(Not run)
```

Index