Package ‘googledrive’

May 5, 2020

Title An Interface to Google Drive

Version 1.0.1

Description Manage Google Drive files from R.

License MIT + file LICENSE

URL https://googledrive.tidyverse.org,
https://github.com/tidyverse/googledrive

BugReports https://github.com/tidyverse/googledrive/issues

Depends R (>= 3.2)

Imports curl (>= 2.8.1), gargle (>= 0.3.1), glue (>= 1.2.0), httr, jsonlite, magrittr, purrr (>= 0.2.3), rlang (>= 0.3.1), tibble (>= 2.0.0), utils, uuid

Suggests covr, dplyr, knitr, rmarkdown, roxygen2, sodium, spelling, testthat (>= 2.1.1)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author Lucy D'Agostino McGowan [aut],
Jennifer Bryan [aut, cre] (<https://orcid.org/0000-0002-6983-2759>),
RStudio [cph, fnd]

Maintainer Jennifer Bryan <jenny@rstudio.com>

Repository CRAN

Date/Publication 2020-05-05 16:10:02 UTC

R topics documented:

`as_dribble` *Coerce to Drive files*

Description

Converts various representations of Google Drive files into a `dribble`, the object used by googledrive to hold Drive file metadata. Files can be specified via:

- File path. File name is an important special case.
- File id. Mark with `as_id()` to distinguish from file path.
- Data frame or `dribble`. Once you've successfully used googledrive to identify the files of interest, you'll have a `dribble`. Pass it into downstream functions.
- List representing Files resource objects. Mostly for internal use.

This is a generic function.

For maximum clarity, get your files into a `dribble` (or capture file id) as early as possible. When specifying via path, it's best to include the trailing slash when you're targetting a folder. If you want the folder `foo`, say foo/, not `foo`.

Some functions, such as `drive_cp()`, `drive_mkdir()`, `drive_mv()`, and `drive_upload()`, can accept the new file or folder name as the last part of `path`, when `name` is not given. But if you say `a/b/c` (no trailing slash) and a folder a/b/c/ already exists, it's unclear what you want. A file named `c` in a/b/ or a file with default name in a/b/c/? You get an error and must make your intent clear.

Usage

```
as_dribble(x, ...)
```

Arguments

`x`	A vector of Drive file paths, a vector of file ids marked with `as_id()`, a list of Files Resource objects, or a suitable data frame.
`...`	Other arguments passed down to methods. (Not used.)

Examples

```
## Not run:
# specify the path (substitute names or paths on your Drive!)
as_dribble("abc")
as_dribble("abc/def")

# specify the file id (substitute a real file id of your own!)
as_dribble(as_id("0B0Gh-SuuA2nTOGZVTXZTREgwZ2M"))

## End(Not run)
```

as_id *Extract and/or mark as file id*

Description

Gets file ids from various inputs and marks them as such, to distinguish them from file names or paths.

This is a generic function.

Usage

```
as_id(x, ...)
```

Arguments

x	A character vector of file or Team Drive ids or URLs, a dribble or a suitable data frame.
...	Other arguments passed down to methods. (Not used.)

Value

A character vector bearing the S3 class `drive_id`.

Examples

```
as_id("123abc")
as_id("https://docs.google.com/spreadsheets/d/qawsedrf16273849/edit#gid=12345")

## Not run:
x <- drive_find(n_max = 3)
as_id(x)

x <- drive_get("foofy")
as_id(x)

x <- team_drive_find("work-stuff")
as_id(x)

## End(Not run)
```

`as_team_drive` *Coerce to Team Drive*

Description

Converts various representations of Team Drive into a `dribble`, the object used by googledrive to hold Drive file metadata. Team Drives can be specified via

- Name.
- Team Drive id. Mark with `as_id()` to distinguish from name.
- Data frame or `dribble` consisting solely of Team Drives.
- List representing Team Drive resource objects. Mostly for internal use.

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

This is a generic function.

Usage

```
as_team_drive(x, ...)
```

Arguments

`x`	A vector of Team Drive names, a vector of Team Drive ids marked with `as_id()`, a list of Team Drive Resource objects, or a suitable data frame.
`...`	Other arguments passed down to methods. (Not used.)

Examples

```
## Not run:
## specify the name
as_team_drive("abc")

## specify the id (substitute one of your own!)
as_team_drive(as_id("0AOPK1X2jaNckUk9PVA"))

## End(Not run)
```

`dribble` *dribble object*

Description

googledrive stores the metadata for one or more Drive files or Team Drives as a `dribble`. It is a "Drive tibble" with one row per file or Team Drive and, at a minimum, these variables:

- `name`: a character variable containing file or Team Drive names
- `id`: a character variable of file or Team Drive ids
- `drive_resource`: a list-column, each element of which is either a Files resource or Team Drive resource object. Note there is no guarantee that all documented fields are always present. We do check if the `kind` field is present and equal to one of `drive#file` or `drive#teamDrive`.

The `dribble` format is handy because it exposes the file name, which is good for humans, but keeps it bundled with the file's unique id and other metadata, which are needed for API calls.

In general, the dribble class will be retained even after subsetting, as long as the required variables are present and of the correct type.

See Also

`as_dribble()`

`dribble-checks` *Check facts about a dribble*

Description

Sometimes you need to check things about a `dribble`‘ or about the files it represents, such as:

- Is it even a dribble?
- Size: Does the dribble hold exactly one file? At least one file? No file?
- File type: Is this file a folder?
- File ownership and access: Is it mine? Published? Shared?

Usage

```
is_dribble(d)

no_file(d)

single_file(d)

some_files(d)
```

```
confirm_dribble(d)

confirm_single_file(d)

confirm_some_files(d)

is_folder(d)

is_native(d)

is_parental(d)

is_mine(d)

is_team_drive(d)

is_team_drivy(d)
```

Arguments

`d`	A `dribble`.

Examples

```
## Not run:
## most of us have multiple files or folders on Google Drive
d <- drive_find()
is_dribble(d)
no_file(d)
single_file(d)
some_files(d)
confirm_single_file(d)
confirm_some_files(d)
is_folder(d)
is_mine(d)

## End(Not run)
```

`drive_about` *Get info on Drive capabilities*

Description

Gets information about the user, the user's Drive, and system capabilities. This function mostly exists to power `drive_user()`, which extracts the most useful information (the information on current user) and prints it nicely.

Usage

```
drive_about()
```

Value

A list representation of a Drive about resource

See Also

Wraps the `about.get` endpoint:

- `https://developers.google.com/drive/v3/reference/about/get`

Examples

```
## Not run:
drive_about()

## explore the names of available Team Drive themes
about <- drive_about()
about[["teamDriveThemes"]] %>%
  purrr::map_chr("id")

## End(Not run)
```

`drive_auth` *Authorize googledrive*

Description

Authorize googledrive to view and manage your Drive files. This function is a wrapper around `gargle::token_fetch()`.

By default, you are directed to a web browser, asked to sign in to your Google account, and to grant googledrive permission to operate on your behalf with Google Drive. By default, these user credentials are cached in a folder below your home directory, ~/.R/gargle/gargle-oauth, from where they can be automatically refreshed, as necessary. Storage at the user level means the same token can be used across multiple projects and tokens are less likely to be synced to the cloud by accident.

If you are interacting with R from a web-based platform, like RStudio Server or Cloud, you need to use a variant of this flow, known as out-of-band auth ("oob"). If this does not happen automatically, you can request it yourself with `use_oob = TRUE` or, more persistently, by setting an option via `options(gargle_oob_default = TRUE)`.

Usage

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

Arguments

`email`	Optional. Allows user to target a specific Google identity. If specified, this is used for token lookup, i.e. to determine if a suitable token is already available in the cache. If no such token is found, `email` is used to pre-select the targetted Google identity in the OAuth chooser. Note, however, that the email associated with a token when it's cached is always determined from the token itself, never from this argument. Use `NA` or `FALSE` to match nothing and force the OAuth dance in the browser. Use `TRUE` to allow email auto-discovery, if exactly one matching token is found in the cache. Defaults to the option named "gargle_oauth_email", retrieved by `gargle::gargle_oauth_email()`.
`path`	JSON identifying the service account, in one of the forms supported for the `txt` argument of `jsonlite::fromJSON()` (typically, a file path or JSON string).
`scopes`	A character vector of scopes to request. Pick from those listed at `https://developers.google.com/identity/protocols/googlescopes`. For certain token flows, the `"https://www.googleapis.com/auth/userinfo.email"` scope is unconditionally included. This grants permission to retrieve the email address associated with a token; gargle uses this to index cached OAuth tokens. This grants no permission to view or send email. It is considered a low value scope and does not appear on the consent screen.
`cache`	Specifies the OAuth token cache. Defaults to the option named "gargle_oauth_cache", retrieved via `gargle::gargle_oauth_cache()`.
`use_oob`	Whether to prefer "out of band" authentication. Defaults to the option named "gargle_oob_default", retrieved via `gargle::gargle_oob_default()`.
`token`	A token with class Token2.0 or an object of httr's class `request`, i.e. a token that has been prepared with `httr::config()` and has a Token2.0 in the `auth_token` component.

Details

Most users, most of the time, do not need to call `drive_auth()` explicitly – it is triggered by the first action that requires authorization. Even when called, the default arguments often suffice. However, when necessary, this function allows the user to explicitly:

- Declare which Google identity to use, via an email address. If there are multiple cached tokens, this can clarify which one to use. It can also force googledrive to switch from one identity to another. If there's no cached token for the email, this triggers a return to the browser to choose the identity and give consent.

- Use a service account token.
- Bring their own Token2.0.
- Specify non-default behavior re: token caching and out-of-bound authentication.

For details on the many ways to find a token, see `gargle::token_fetch()`. For deeper control over auth, use `drive_auth_configure()` to bring your own OAuth app or API key. Read more about gargle options, see gargle::gargle_options.

See Also

Other auth functions: `drive_auth_configure`(), `drive_deauth`()

Examples

```
## Not run:
## load/refresh existing credentials, if available
## otherwise, go to browser for authentication and authorization
drive_auth()

## see user associated with current token
drive_user()

## force use of a token associated with a specific email
drive_auth(email = "jenny@example.com")
drive_user()

## force a menu where you can choose from existing tokens or
## choose to get a new one
drive_auth(email = NA)

## use a 'read only' scope, so it's impossible to edit or delete files
drive_auth(
  scopes = "https://www.googleapis.com/auth/drive.readonly"
)

## use a service account token
drive_auth(path = "foofy-83ee9e7c9c48.json")

## End(Not run)
```

`drive_auth_configure` *Edit and view auth configuration*

Description

These functions give more control over and visibility into the auth configuration than `drive_auth()` does. `drive_auth_configure()` lets the user specify their own:

- OAuth app, which is used when obtaining a user token.

- API key. If googledrive is de-authorized via `drive_deauth()`, all requests are sent with an API key in lieu of a token. See the vignette How to get your own API credentials for more.

If the user does not configure these settings, internal defaults are used. `drive_oauth_app()` and `drive_api_key()` retrieve the currently configured OAuth app and API key, respectively.

Usage

```
drive_auth_configure(app, path, api_key)

drive_api_key()

drive_oauth_app()
```

Arguments

`app`	OAuth app, in the sense of `httr::oauth_app()`.
`path`	JSON downloaded from Google Cloud Platform Console, containing a client id (aka key) and secret, in one of the forms supported for the `txt` argument of `jsonlite::fromJSON()` (typically, a file path or JSON string).
`api_key`	API key.

Value

- `drive_auth_configure()`: An object of R6 class gargle::AuthState, invisibly.
- `drive_oauth_app()`: the current user-configured `httr::oauth_app()`.
- `drive_api_key()`: the current user-configured API key.

See Also

Other auth functions: `drive_auth()`, `drive_deauth()`

Examples

```
# see and store the current user-configured OAuth app (probaby `NULL`)
(original_app <- drive_oauth_app())

# see and store the current user-configured API key (probaby `NULL`)
(original_api_key <- drive_api_key())

if (require(httr)) {
  # bring your own app via client id (aka key) and secret
  google_app <- httr::oauth_app(
    "my-awesome-google-api-wrapping-package",
    key = "123456789.apps.googleusercontent.com",
    secret = "abcdefghijklmnopqrstuvwxyz"
  )
  google_key <- "the-key-I-got-for-a-google-API"
  drive_auth_configure(app = google_app, api_key = google_key)

  # confirm the changes
  drive_oauth_app()
  drive_api_key()
}

## Not run:
## bring your own app via JSON downloaded from Google Developers Console
drive_auth_configure(
  path = "/path/to/the/JSON/you/downloaded/from/google/dev/console.json"
)

## End(Not run)

# restore original auth config
drive_auth_configure(app = original_app, api_key = original_api_key)
```

drive_browse *Visit Drive file in browser*

Description

Visits a file on Google Drive in your default browser.

Usage

```
drive_browse(file = .Last.value)
```

Arguments

`file`	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.

Value

Character vector of file hyperlinks, from `drive_link()`, invisibly.

Examples

```
## Not run:
drive_find(n_max = 1) %>% drive_browse()

## End(Not run)
```

`drive_cp` *Copy a Drive file*

Description

Copies an existing Drive file into a new file id.

Usage

```
drive_cp(file, path = NULL, name = NULL, ..., overwrite = NA, verbose = TRUE)
```

Arguments

`file`	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.
`path`	Specifies target destination for the new file on Google Drive. Can be an actual path (character), a file id marked with `as_id()`, or a `dribble`. If specified as an actual path, it is best to explicitly indicate if it's a folder by including a trailing slash, since it cannot always be worked out from the context of the call. Defaults to "Copy of `FILE-NAME`".
`name`	Character, new file name if not specified as part of `path`. This will force `path` to be treated as a folder, even if it is character and lacks a trailing slash. Defaults to "Copy of `FILE-NAME`".
`...`	Named parameters to pass along to the Drive API. Has the tidy dots semantics that come from using `rlang::list2()`. You can affect the metadata of the target file by specifying properties of the Files resource via `...`. Read the "Request body" section of the Drive API docs for the associated endpoint to learn about relevant parameters.
`overwrite`	Logical, indicating whether to check for a pre-existing file at the targetted "filepath". The quotes around "filepath" refer to the fact that Drive does not impose a 1-to-1 relationship between filepaths and files, like a typical file system; read more about that in `drive_get()`. • `NA` (default): Just do the operation, even if it results in multiple files with the same filepath. • `TRUE`: Check for a pre-existing file at the filepath. If there is zero or one, move a pre-existing file to the trash, then carry on. Note that the new file does not inherit any properties from the old one, such as sharing or publishing settings. It will have a new file ID. An error is thrown if two or more pre-existing files are found. • `FALSE`: Error if there is any pre-existing file at the filepath. Note that existence checks, based on filepath, are expensive operations, i.e. they require additional API calls.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `files.copy` endpoint:

- https://developers.google.com/drive/v3/reference/files/copy

Examples

```
## Not run:
## Create a file to copy
file <- drive_upload(drive_example("chicken.txt"), "chicken-cp.txt")

## Make a "Copy of" copy in same folder as the original
drive_cp("chicken-cp.txt")

## Make an explicitly named copy in same folder as the original
drive_cp("chicken-cp.txt", "chicken-cp-two.txt")

## Make an explicitly named copy in a different folder
folder <- drive_mkdir("new-folder")
drive_cp("chicken-cp.txt", path = folder, name = "chicken-cp-three.txt")

## Make an explicitly named copy and star it.
## The starring is an example of providing metadata via `...`.
## `starred` is not an actual argument to `drive_cp()`,
## it just gets passed through to the API.
drive_cp("chicken-cp.txt", name = "chicken-cp-starred.txt", starred = TRUE)

## `overwrite = FALSE` errors if file already exists at target filepath
## THIS WILL ERROR!
drive_cp("chicken-cp.txt", name = "chicken-cp.txt", overwrite = FALSE)

## `overwrite = TRUE` moves an existing file to trash, then proceeds
drive_cp("chicken-cp.txt", name = "chicken-cp.txt", overwrite = TRUE)

## Behold all of our copies!
drive_find("chicken-cp")

## Delete all of our copies and the new folder!
drive_find("chicken-cp") %>% drive_rm()
drive_rm(folder)

## upload a csv file to copy
csv_file <- drive_upload(drive_example("chicken.csv"))

## copy AND AT THE SAME TIME convert it to a Google Sheet
chicken_sheet <- drive_cp(
  csv_file,
  name = "chicken-cp",
  mime_type = drive_mime_type("spreadsheet")
)

## go see the new Sheet in the browser
## drive_browse(chicken_sheet)

## clean up
drive_rm(csv_file, chicken_sheet)

## End(Not run)
```

`drive_create` *Create a new blank Drive file*

Description

Creates a new blank Drive file. Note there are better options for these special cases:

- Creating a folder? Use `drive_mkdir()`.
- Want to upload existing local content into a new Drive file? Use `drive_upload()`.

Usage

```
drive_create(
  name,
  path = NULL,
  type = NULL,
  ...,
  overwrite = NA,
  verbose = TRUE
)
```

Arguments

`name`	Name for the new file or, optionally, a path that specifies an existing parent folder, as well as the new file name.
`path`	Target destination for the new item, i.e. a folder or a Team Drive. Can be given as an actual path (character), a file id or URL marked with `as_id()`, or a `dribble`. Defaults to your "My Drive" root folder.
`type`	Character. Create a blank Google Doc, Sheet or Slides by setting `type` to `document`, `spreadsheet`, or `presentation`, respectively. All non-`NULL` values for `type` are pre-processed with `drive_mime_type()`.
`...`	Named parameters to pass along to the Drive API. Has the tidy dots semantics that come from using `rlang::list2()`. You can affect the metadata of the target file by specifying properties of the Files resource via `...`. Read the "Request body" section of the Drive API docs for the associated endpoint to learn about relevant parameters.

`overwrite`	Logical, indicating whether to check for a pre-existing file at the targetted "filepath". The quotes around "filepath" refer to the fact that Drive does not impose a 1-to-1 relationship between filepaths and files, like a typical file system; read more about that in `drive_get()`. • `NA` (default): Just do the operation, even if it results in multiple files with the same filepath. • `TRUE`: Check for a pre-existing file at the filepath. If there is zero or one, move a pre-existing file to the trash, then carry on. Note that the new file does not inherit any properties from the old one, such as sharing or publishing settings. It will have a new file ID. An error is thrown if two or more pre-existing files are found. • `FALSE`: Error if there is any pre-existing file at the filepath. Note that existence checks, based on filepath, are expensive operations, i.e. they require additional API calls.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `files.create` endpoint:

- https://developers.google.com/drive/v3/reference/files/create

Examples

```
## Not run:
## Create a blank Google Doc named 'WordStar' in
## your 'My Drive' root folder and star it
wordstar <- drive_create("WordStar", type = "document", starred = TRUE)

## is 'WordStar' really starred? YES
purrr::pluck(wordstar, "drive_resource", 1, "starred")

## Create a blank Google Slides presentation in
## the root folder, and set its description
execuvision <- drive_create(
  "ExecuVision",
  type = "presentation",
  description = "deeply nested bullet lists FTW"
)

## Did we really set the description? YES
purrr::pluck(execuvision, "drive_resource", 1, "description")

## check out the new presentation
drive_browse(execuvision)
```

```
## Create folder 'b4xl' in the root folder,
## then create an empty new Google Sheet in it
b4xl <- drive_mkdir("b4xl")
drive_create("VisiCalc", path = b4xl, type = "spreadsheet")

## Another way to create a Google Sheet in the folder 'b4xl'
drive_create("b4xl/SuperCalc", type = "spreadsheet")

## Another way to create a new file in a folder,
## this time specifying parent `path` as a character
drive_create("Lotus 1-2-3", path = "b4xl", type = "spreadsheet")

## `overwrite = FALSE` errors if file already exists at target filepath
## THIS WILL ERROR!
drive_create("VisiCalc", path = b4xl, overwrite = FALSE)

## `overwrite = TRUE` moves an existing file to trash, then proceeds
drive_create("VisiCalc", path = b4xl, overwrite = TRUE)

## clean up
drive_rm(wordstar, b4xl, execuvision)

## End(Not run)
```

drive_deauth *Suspend authorization*

Description

Put googledrive into a de-authorized state. Instead of sending a token, googledrive will send an API key. This can be used to access public resources for which no Google sign-in is required. This is handy for using googledrive in a non-interactive setting to make requests that do not require a token. It will prevent the attempt to obtain a token interactively in the browser. The user can configure their own API key via `drive_auth_configure()` and retrieve that key via `drive_api_key()`. In the absence of a user-configured key, a built-in default key is used.

Usage

```
drive_deauth()
```

See Also

Other auth functions: `drive_auth_configure`(), `drive_auth`()

Examples

```
## Not run:
drive_deauth()
drive_user()
```

```
public_file <-
  drive_get(as_id("1Hj-k7NpPSyeOR3R7j4KuWnru6kZaqqOAE8_db5gowIM"))
drive_download(public_file)

## End(Not run)
```

drive_download *Download a Drive file*

Description

This function downloads a file from Google Drive. Native Google file types, such as Google Docs, Google Sheets, and Google Slides, must be exported to a conventional local file type. This can be specified:

- explicitly via `type`
- implicitly via the file extension of `path`
- not at all, i.e. rely on default built into googledrive

To see what export file types are even possible, see the Drive API documentation. Returned dribble contains local path to downloaded file in `local_path`.

Usage

```
drive_download(
  file,
  path = NULL,
  type = NULL,
  overwrite = FALSE,
  verbose = TRUE
)
```

Arguments

`file`	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.
`path`	Character. Path for output file. If absent, the default file name is the file's name on Google Drive and the default location is working directory, possibly with an added file extension.
`type`	Character. Only consulted if `file` is a native Google file. Specifies the desired type of the downloaded file. Will be processed via `drive_mime_type()`, so either a file extension like `"pdf"` or a full MIME type like `"application/pdf"` is acceptable.
`overwrite`	A logical scalar. If local `path` already exists, do you want to overwrite it?
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
## Upload a csv file into a Google Sheet
file <- drive_upload(
  drive_example("chicken.csv"),
  type = "spreadsheet"
)

## Download Sheet as csv, explicit type
(downloaded_file <- drive_download(file, type = "csv"))

## See local path to new file
downloaded_file$local_path

## Download as csv, type implicit in file extension
drive_download(file, path = "my_csv_file.csv")

## Download with default name and type (xlsx)
drive_download(file)

## Clean up
unlink(c("chicken.csv", "chicken.csv.xlsx", "my_csv_file.csv"))
drive_rm(file)

## End(Not run)
```

`drive_empty_trash` *Empty Drive Trash*

Description

Caution, this will permanently delete files in your Drive trash.

Usage

```
drive_empty_trash(verbose = TRUE)
```

Arguments

`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

drive_endpoints *List Drive endpoints*

Description

The googledrive package stores a named list of Drive API v3 endpoints (or "methods", using Google's vocabulary) internally and these functions expose this data.

- `drive_endpoint()` returns one endpoint, i.e. it uses `[[`.
- `drive_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 Drive API v3 Discovery Document using the approach described in the Discovery Documents section of the gargle vignette Request helper functions.

Usage

```
drive_endpoints(i = NULL)

drive_endpoint(i)
```

Arguments

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

Value

One or more of the Drive API v3 endpoints that are used internally by googledrive.

Examples

```
str(head(drive_endpoints(), 3), max.level = 2)
drive_endpoint("drive.files.delete")
drive_endpoint(4)
```

`drive_example` *Get path to example file*

Description

googledrive comes bundled with a few small files to use in examples. This function make them easy to access.

Usage

```
drive_example(path = NULL)
```

Arguments

`path`	Name of file. If `NULL`, the example files will be listed.

Examples

```
drive_example()
drive_example("chicken.jpg")
```

`drive_extension` *Lookup extension from MIME type*

Description

This is a helper to determinine which extension should be used for a file. Two types of input are acceptable:

- MIME types accepted by Google Drive.
- File extensions, such as "pdf", "csv", etc. (these are simply passed through).

Usage

```
drive_extension(type = NULL)
```

Arguments

`type`	Character. MIME type or file extension.

Value

Character. File extension.

Examples

```
## get the extension for mime type image/jpeg
drive_extension("image/jpeg")

## it's vectorized
drive_extension(c("text/plain", "pdf", "image/gif"))
```

`drive_fields` *Request partial resources*

Description

You may be able to improve the performance of your API calls by requesting only the metadata that you actually need. This function is primarily for internal use and is currently focused on the Files resource. Note that high-level googledrive functions assume that the `name`, `id`, and `kind` fields are included, at a bare minimum. Assuming that `resource = "files"` (the default), input provided via `fields` is checked for validity against the known field names and the validated fields are returned. To see a tibble containing all possible fields and a short description of each, call `drive_fields(expose())`.

`prep_fields()` prepares fields for inclusion as query parameters.

Usage

```
drive_fields(fields = NULL, resource = "files")

prep_fields(fields, resource = "files")
```

Arguments

`fields`	Character vector of field names. If `resource = "files"`, they are checked for validity. Otherwise, they are passed through.
`resource`	Character, naming the API resource of interest. Currently, only the Files resource is anticipated.

Value

`drive_fields()`: Character vector of field names. `prep_fields()`: a string.

See Also

Working with partial resources, in the Drive API documentation.

Examples

```
## get a tibble of all fields for the Files resource + indicator of defaults
drive_fields(expose())

## invalid fields are removed and throw warning
drive_fields(c("name", "parents", "ownedByMe", "pancakes!"))

## prepare fields for query
prep_fields(c("name", "parents", "kind"))
```

`drive_find` *Find files on Google Drive*

Description

This is the closest googledrive function to what you can do at https://drive.google.com: by default, you just get a listing of your files. You can also search in various ways, e.g., filter by file type or ownership or even work with Team Drive files, if you have access. This is a very powerful function. Together with the more specific `drive_get()`, this is the main way to identify files to target for downstream work.

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
drive_find(
  pattern = NULL,
  trashed = FALSE,
  type = NULL,
  n_max = Inf,
  team_drive = NULL,
  corpus = NULL,
  ...,
  verbose = TRUE
)
```

Arguments

`pattern`	Character. If provided, only the items whose names match this regular expression are returned. This is implemented locally on the results returned by the API.
`trashed`	Logical. Whether to search files that are not in the trash (`trashed = FALSE`, the default), only files that are in the trash (`trashed = TRUE`), or to search regardless of trashed status (`trashed = NA`).
`type`	Character. If provided, only files of this type will be returned. Can be anything that `drive_mime_type()` knows how to handle. This is processed by googledrive and sent as a query parameter.

n_max	Integer. An upper bound on the number of items to return. This applies to the results requested from the API, which may be further filtered locally, via the pattern argument.
team_drive	Anything that identifies one specific Team Drive: its name, its id or URL marked with as_id(), or a dribble. Is pre-processed with as_team_drive(). Read more about Team Drives.
corpus	Character, specifying the search collection. Only relevant in the Team Drives context. If specified, must be one of "user", "all", or "domain". Read more about Team Drives.
...	Other parameters to pass along in the request. The most likely candidate is q. See below and the API's Search for files and folders guide.
verbose	Logical, indicating whether to print informative messages (default TRUE).

Value

An object of class dribble, a tibble with one row per item.

File type

The type argument is pre-processed with drive_mime_type(), so you can use a few shortcuts and file extensions, in addition to full-blown MIME types. googledrive forms a search clause to pass to q.

Search parameters

Do advanced search on file properties by providing search clauses to the q parameter that is passed to the API via .... Multiple q clauses or vector-valued q are combined via 'and'.

Trash

By default, drive_find() sets trashed = FALSE and does not include files in the trash. Literally, it adds q = "trashed = false" to the query. To search *only* the trash, set trashed = TRUE. To see files regardless of trash status, set trashed = NA, which adds q = "(trashed = true or trashed = false)" to the query.

Sort order

By default, drive_find() sends orderBy = "recency desc", so the top files in your result have high "recency" (whatever that means). To suppress sending orderBy at all, do drive_find(orderBy = NULL). The orderBy parameter accepts sort keys in addition to recency, which are documented in the files.list endpoint. googledrive translates a snake_case specification of order_by into the lowerCamel form, orderBy.

Team Drives

If you have access to Team Drives, you'll know. Use team_drive or corpus to search one or more Team Drives or a domain. See Access Team Drives for more.

See Also

Wraps the `files.list` endpoint:

- https://developers.google.com/drive/v3/reference/files/list

Helpful resource for forming your own queries:

- https://developers.google.com/drive/api/v3/search-files

Examples

```
## Not run:
# list "My Drive" w/o regard for folder hierarchy
drive_find()

# filter for folders, the easy way and the hard way
drive_find(type = "folder")
drive_find(q = "mimeType = 'application/vnd.google-apps.folder'")

# filter for Google Sheets, the easy way and the hard way
drive_find(type = "spreadsheet")
drive_find(q = "mimeType='application/vnd.google-apps.spreadsheet'")

# files whose names match a regex
drive_find(pattern = "jt")

# search for files located directly in your root folder
drive_find(q = "'root' in parents")
# FYI: this is equivalent to
drive_ls("~/")

# control page size or cap the number of files returned
drive_find(pageSize = 50)
# all params passed through `...` can be camelCase or snake_case
drive_find(page_size = 50)
drive_find(n_max = 58)
drive_find(page_size = 5, n_max = 15)

# various ways to specify q search clauses
# multiple q's
drive_find(q = "name contains 'TEST'",
           q = "modifiedTime > '2017-07-21T12:00:00'")
# vector q
drive_find(q = c("starred = true", "visibility = 'anyoneWithLink'"))

# default `trashed = FALSE` excludes files in the trash
# `trashed = TRUE` consults ONLY file in the trash
drive_find(trashed = TRUE)
# `trashed = NA` disregards trash status completely
drive_find(trashed = NA)

# suppress the default sorting on recency
drive_find(order_by = NULL, n_max = 5)
```

```
# sort on various keys
drive_find(order_by = "quotaBytesUsed", n_max = 5)
drive_find(order_by = "modifiedByMeTime", n_max = 5)

## End(Not run)
```

drive_get *Get Drive files by path or id*

Description

Retrieve metadata for files specified via path or via file id.

If the files are specified via `path`, the returned `dribble` will include a `path` variable. To add path information to any `dribble` that lacks it, use `drive_reveal()`, e.g., `drive_reveal(d, "path")`. If you want to list the contents of a folder, use `drive_ls()`. For general searching, use `drive_find()`.

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
drive_get(
  path = NULL,
  id = NULL,
  team_drive = NULL,
  corpus = NULL,
  verbose = TRUE
)
```

Arguments

`path`	Character vector of path(s) to get. Use a trailing slash to indicate explicitly that a path is a folder, which can disambiguate if there is a file of the same name (yes this is possible on Drive!). If `path` appears to contain Drive URLs or is explicitly marked with `as_id()`, it is treated as if it was provided via the `id` argument.
`id`	Character vector of Drive file ids or URLs (it is first processed with `as_id()`). If both `path` and `id` are non-`NULL`, `id` is silently ignored.
`team_drive`	Anything that identifies one specific Team Drive: its name, its id or URL marked with `as_id()`, or a `dribble`. Is pre-processed with `as_team_drive()`. Read more about Team Drives.
`corpus`	Character, specifying the search collection. Only relevant in the Team Drives context. If specified, must be one of `"user"`, `"all"`, or `"domain"`. Read more about Team Drives.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class dribble, a tibble with one row per item.

Special considerations for paths

Note that Google Drive does NOT behave like your local file system:

- File and folder names need not be unique, even at a given level of the hierarchy. A single name or file path can be associated with multiple files (or zero or exactly one).
- A file can have more than one direct parent. This implies that a single file can be represented by multiple paths.

Bottom line: Do not assume there is a one-to-one relationship between file name or path and a Drive file or folder. This implies the length of the input (i.e. the number of input paths or the number of rows in a dribble) will not necessarily equal the number rows in the output.

See Also

Wraps the `files.get` endpoint and, if you specify files by name or path, also calls `files.list`:

- https://developers.google.com/drive/v3/reference/files/get
- https://developers.google.com/drive/v3/reference/files/list

Examples

```
## Not run:
## get info about your "My Drive" root folder
drive_get("~/")
## the API reserves the file id "root" for your root folder
drive_get(id = "root")
drive_get(id = "root") %>% drive_reveal("path")

## The examples below are indicative of correct syntax.
## But note these will generally result in an error or a
## 0-row dribble, unless you replace the inputs with paths
## or file ids that exist in your Drive.

## multiple names
drive_get(c("abc", "def"))

## multiple names, one of which must be a folder
drive_get(c("abc", "def/"))

## query by file id(s)
drive_get(id = "abcdefgeh123456789")
drive_get(as_id("abcdefgeh123456789"))
drive_get(id = c("abcdefgh123456789", "jklmnopq123456789"))

## apply to a browser URL for, e.g., a Google Sheet
my_url <- "https://docs.google.com/spreadsheets/d/FILE_ID/edit#gid=SHEET_ID"
drive_get(my_url)
drive_get(as_id(my_url))
drive_get(id = my_url)

## access the Team Drive named "foo"
## team_drive params must be specified if getting by path
foo <- team_drive_get("foo")
drive_get(c("this.jpg", "that-file"), team_drive = foo)
## team_drive params are not necessary if getting by id
drive_get(as_id("123456789"))

## search all Team Drives and other files user has accessed
drive_get(c("this.jpg", "that-file"), corpus = "all")

## End(Not run)
```

`drive_has_token` *Is there a token on hand?*

Description

Reports whether googledrive has stored a token, ready for use in downstream requests.

Usage

```
drive_has_token()
```

Value

Logical.

See Also

Other low-level API functions: `drive_token`(), `request_generate`(), `request_make`()

Examples

```
drive_has_token()
```

`drive_link` *Retrieve Drive file links*

Description

Returns the `"webViewLink"` for one or more files, which is the "link for opening the file in a relevant Google editor or viewer in a browser".

Usage

```
drive_link(file)
```

Arguments

`file`	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.

Value

Character vector of file hyperlinks.

Examples

```
## Not run:
## get a few files into a dribble
three_files <- drive_find(n_max = 3)

## get their browser links
drive_link(three_files)

## End(Not run)
```

`drive_ls` *List contents of a folder or Team Drive*

Description

List the contents of a folder or Team Drive, recursively or not. This is a thin wrapper around `drive_find()`, that simply adds one constraint: the search is limited to direct or indirect children of `path`.

Usage

```
drive_ls(path = NULL, ..., recursive = FALSE)
```

Arguments

`path`	Specifies a single folder on Google Drive whose contents you want to list. Can be an actual path (character), a file id or URL marked with `as_id()`, or a `dribble`. If it is a Team Drive or is a folder on a Team Drive, it must be passed as a `dribble`.
`...`	Any parameters that are valid for `drive_find()`.
`recursive`	Logical, indicating if you want only direct children of `path` (`recursive` = `FALSE`, the default) or all children, including indirect (`recursive` = `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
## get contents of the folder 'abc' (non-recursive)
drive_ls("abc")

## get contents of folder 'abc' whose names contain the letters 'def'
drive_ls(path = "abc", pattern = "def")

## get all Google spreadsheets in folder 'abc'
## whose names contain the letters 'def'
drive_ls(path = "abc", pattern = "def", type = "spreadsheet")

## get all the files below 'abc', recursively, that are starred
drive_ls(path = "abc", q = "starred = true", recursive = TRUE)

## End(Not run)
```

`drive_mime_type` *Lookup MIME type*

Description

This is a helper to determine which MIME type should be used for a file. Three types of input are acceptable:

- Native Google Drive file types. Important examples:
  - "document" for Google Docs
  - "folder" for folders
  - "presentation" for Google Slides
  - "spreadsheet" for Google Sheets
- File extensions, such as "pdf", "csv", etc.
- MIME types accepted by Google Drive (these are simply passed through).

Usage

```
drive_mime_type(type = NULL)
```

Arguments

`type`	Character. Google Drive file type, file extension, or MIME type. Pass the sentinel `expose()` if you want to get the full table used for validation and lookup, i.e. all MIME types known to be relevant to the Drive API.

Value

Character. MIME type.

Examples

```
## get the mime type for Google Spreadsheets
drive_mime_type("spreadsheet")

## get the mime type for jpegs
drive_mime_type("jpeg")

## it's vectorized
drive_mime_type(c("presentation", "pdf", "image/gif"))

## see the internal tibble of MIME types known to the Drive API
drive_mime_type(expose())
```

`drive_mkdir` *Create a Drive folder*

Description

Creates a new Drive folder. To update the metadata of an existing Drive file, including a folder, use `drive_update()`.

Usage

```
drive_mkdir(
  name,
  path = NULL,
  ...,
  overwrite = NA,
  parent = "DEPRECATED",
  verbose = TRUE
)
```

Arguments

`name`	Name for the new folder or, optionally, a path that specifies an existing parent folder, as well as the new name.
`path`	Target destination for the new item, i.e. a folder or a Team Drive. Can be given as an actual path (character), a file id or URL marked with `as_id()`, or a `dribble`. Defaults to your "My Drive" root folder.
`...`	Named parameters to pass along to the Drive API. Has the tidy dots semantics that come from using `rlang::list2()`. You can affect the metadata of the target file by specifying properties of the Files resource via .... Read the "Request body" section of the Drive API docs for the associated endpoint to learn about relevant parameters.
`overwrite`	Logical, indicating whether to check for a pre-existing file at the targetted "filepath". The quotes around "filepath" refer to the fact that Drive does not impose a 1-to-1 relationship between filepaths and files, like a typical file system; read more about that in `drive_get()`. • `NA` (default): Just do the operation, even if it results in multiple files with the same filepath. • `TRUE`: Check for a pre-existing file at the filepath. If there is zero or one, move a pre-existing file to the trash, then carry on. Note that the new file does not inherit any properties from the old one, such as sharing or publishing settings. It will have a new file ID. An error is thrown if two or more pre-existing files are found. • `FALSE`: Error if there is any pre-existing file at the filepath. Note that existence checks, based on filepath, are expensive operations, i.e. they require additional API calls.
`parent`	DEPRECATED. Use the `path` argument for this now, which is more consistent with other functions in googledrive.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `files.create` endpoint:

- `https://developers.google.com/drive/v3/reference/files/create`

Examples

```
## Not run:
## Create folder named 'ghi', then another below named it 'jkl' and star it
ghi <- drive_mkdir("ghi")
jkl <- drive_mkdir("ghi/jkl", starred = TRUE)

## is 'jkl' really starred? YES
```

```
purrr::pluck(jkl, "drive_resource", 1, "starred")

## Another way to create folder 'mno' in folder 'ghi'
drive_mkdir("mno", path = "ghi")

## Yet another way to create a folder named 'pqr' in folder 'ghi',
## this time with parent folder stored in a dribble,
## and setting the new folder's description
pqr <- drive_mkdir("pqr", path = ghi, description = "I am a folder")

## Did we really set the description? YES
purrr::pluck(pqr, "drive_resource", 1, "description")

## `overwrite = FALSE` errors if something already exists at target filepath
## THIS WILL ERROR!
drive_create("name-squatter", path = ghi)
drive_mkdir("name-squatter", path = ghi, overwrite = FALSE)

## `overwrite = TRUE` moves the existing item to trash, then proceeds
drive_mkdir("name-squatter", path = ghi, overwrite = TRUE)

## clean up
drive_rm(ghi)

## End(Not run)
```

drive_mv *Move a Drive file*

Description

Move a Drive file to a different folder, give it a different name, or both. Note that folders on Google Drive are not like folders on your local filesystem. They are more like a label, which implies that a Drive file can have multiple folders as direct parent! However, most people still use and think of them like "regular" folders. When we say "move a Drive file", it actually means: "add a new folder to this file's parents and remove the old one".

Usage

```
drive_mv(file, path = NULL, name = NULL, overwrite = NA, verbose = TRUE)
```

Arguments

file	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.
path	Specifies target destination for the new file on Google Drive. Can be an actual path (character), a file id marked with `as_id()`, or a `dribble`. If specified as an actual path, it is best to explicitly indicate if it's a folder by including a trailing slash, since it cannot always be worked out from the context of the call. Defaults to current name.

`name`	Character, new file name if not specified as part of `path`. This will force `path` to be treated as a folder, even if it is character and lacks a trailing slash. Defaults to current name.
`overwrite`	Logical, indicating whether to check for a pre-existing file at the targetted "filepath". The quotes around "filepath" refer to the fact that Drive does not impose a 1-to-1 relationship between filepaths and files, like a typical file system; read more about that in `drive_get()`. • `NA` (default): Just do the operation, even if it results in multiple files with the same filepath. • `TRUE`: Check for a pre-existing file at the filepath. If there is zero or one, move a pre-existing file to the trash, then carry on. Note that the new file does not inherit any properties from the old one, such as sharing or publishing settings. It will have a new file ID. An error is thrown if two or more pre-existing files are found. • `FALSE`: Error if there is any pre-existing file at the filepath. Note that existence checks, based on filepath, are expensive operations, i.e. they require additional API calls.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
## create a file to move
file <- drive_upload(drive_example("chicken.txt"), "chicken-mv.txt")

## rename it, but leave in current folder (root folder, in this case)
file <- drive_mv(file, "chicken-mv-renamed.txt")

## create a folder to move the file into
folder <- drive_mkdir("mv-folder")

## move the file and rename it again,
## specify destination as a dribble
file <- drive_mv(file, path = folder, name = "chicken-mv-re-renamed.txt")

## verify renamed file is now in the folder
drive_ls(folder)

## move the file back to root folder
file <- drive_mv(file, "~/")

## move it again
## specify destination as path with trailing slash
## to ensure we get a move vs. renaming it to "mv-folder"
file <- drive_mv(file, "mv-folder/")
```

```
## `overwrite = FALSE` errors if something already exists at target filepath
## THIS WILL ERROR!
drive_create("name-squatter", path = "~/")
drive_mv(file, path = "~/", name = "name-squatter", overwrite = FALSE)

## `overwrite = TRUE` moves the existing item to trash, then proceeds
drive_mv(file, path = "~/", name = "name-squatter", overwrite = TRUE)

## Clean up
drive_rm(file, folder)

## End(Not run)
```

`drive_publish` *Publish native Google files*

Description

Publish (or un-publish) native Google files to the web. Native Google files include Google Docs, Google Sheets, and Google Slides. The returned `dribble` will have extra columns, `published` and `revisions_resource`. Read more in `drive_reveal()`.

Usage

```
drive_publish(file, ..., verbose = TRUE)

drive_unpublish(file, ..., verbose = TRUE)
```

Arguments

`file`	Something that identifies the file(s) of interest on your Google Drive. Can be a character vector of names/paths, a character vector of file ids or URLs marked with `as_id()`, or a `dribble`.
`...`	Name-value pairs to add to the API request body (see API docs linked below for details). For `drive_publish()`, we include `publishAuto = TRUE` and `publishedOutsideDomain = TRUE`, if user does not specify other values.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `revisions.update` endpoint:

- `https://developers.google.com/drive/v3/reference/revisions/update`

Examples

```
## Not run:
## Upload file to publish
file <- drive_upload(
  drive_example("chicken.csv"),
  type = "spreadsheet"
  )

## Publish file
file <- drive_publish(file)
file$published

## Unpublish file
file <- drive_unpublish(file)
file$published

## Clean up
drive_rm(file)

## End(Not run)
```

---

drive_put *PUT new media into a Drive file*

Description

PUTs new media into a Drive file, in the HTTP sense: if the file already exists, we replace its content and we create a new file, otherwise. This is a convenience wrapper around `drive_upload()` and `drive_update()`. In pseudo-code:

```
target_filepath <- <determined from `path`, `name`, and `media`>
hits <- <get all Drive files at target_filepath>
if (no hits) {
  drive_upload(media, path, name, type, ..., verbose)
} else if (exactly 1 hit) {
  drive_update(hit, media, ..., verbose)
} else {
  ERROR
}
```

Usage

```
drive_put(media, path = NULL, name = NULL, ..., type = NULL, verbose = TRUE)
```

Arguments

`media`	Character, path to the local file to upload.

`path`	Specifies target destination for the new file on Google Drive. Can be an actual path (character), a file id marked with `as_id()`, or a `dribble`. If specified as an actual path, it is best to explicitly indicate if it's a folder by including a trailing slash, since it cannot always be worked out from the context of the call. Will default to its local name.
`name`	Character, new file name if not specified as part of `path`. This will force `path` to be treated as a folder, even if it is character and lacks a trailing slash. Will default to its local name.
`...`	Named parameters to pass along to the Drive API. Has the tidy dots semantics that come from using `rlang::list2()`. You can affect the metadata of the target file by specifying properties of the Files resource via `...`. Read the "Request body" section of the Drive API docs for the associated endpoint to learn about relevant parameters.
`type`	Character. If `type = NULL`, a MIME type is automatically determined from the file extension, if possible. If the source file is of a suitable type, you can request conversion to Google Doc, Sheet or Slides by setting `type` to `document`, `spreadsheet`, or `presentation`, respectively. All non-`NULL` values for `type` are pre-processed with `drive_mime_type()`.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
# create a local file to work with
local_file <- tempfile("drive_put_", fileext = ".txt")
writeLines(c("beginning", "middle"), local_file)

# PUT to a novel filepath --> drive_put() delegates to drive_upload()
file <- drive_put(local_file)

# update the local file
cat("end", file = local_file, sep = "\n", append = TRUE)

# PUT again --> drive_put() delegates to drive_update()
file <- drive_put(local_file)

# create a second file at this filepath
file2 <- drive_create(basename(local_file))

# PUT again --> ERROR
drive_put(local_file)

# clean-up
drive_find("drive_put_.+[.]txt") %>% drive_rm()
unlink(local_file)
```

```
## End(Not run)
```

`drive_rename` *Rename a Drive file*

Description

This is a wrapper for `drive_mv()` that only renames a file. If you would like to rename AND move the file, see `drive_mv()`.

Usage

```
drive_rename(file, name = NULL, overwrite = NA, verbose = TRUE)
```

Arguments

`file`	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.
`name`	Character. Name you would like the file to have.
`overwrite`	Logical, indicating whether to check for a pre-existing file at the targetted "filepath". The quotes around "filepath" refer to the fact that Drive does not impose a 1-to-1 relationship between filepaths and files, like a typical file system; read more about that in `drive_get()`. • `NA` (default): Just do the operation, even if it results in multiple files with the same filepath. • `TRUE`: Check for a pre-existing file at the filepath. If there is zero or one, move a pre-existing file to the trash, then carry on. Note that the new file does not inherit any properties from the old one, such as sharing or publishing settings. It will have a new file ID. An error is thrown if two or more pre-existing files are found. • `FALSE`: Error if there is any pre-existing file at the filepath. Note that existence checks, based on filepath, are expensive operations, i.e. they require additional API calls.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
## Create a file to rename
file <- drive_create("file-to-rename")

## Rename it
```

```
file <- drive_rename(file, name = "renamed-file")

## `overwrite = FALSE` errors if something already exists at target filepath
## THIS WILL ERROR!
drive_create("name-squatter")
drive_rename(file, name = "name-squatter", overwrite = FALSE)

## `overwrite = TRUE` moves the existing item to trash, then proceeds
file <- drive_rename(file, name = "name-squatter", overwrite = TRUE)

## Clean up
drive_rm(file)

## End(Not run)
```

`drive_reveal` *Add column(s) with new information*

Description

`drive_reveal()` adds extra information about your Drive files that is not automatically present in the default `dribble` produced by googledrive. Why is this info not always present?

1. You don't always care about it.
2. It may require calling different endpoints in the Drive API. Example: determining if a file has been "published on the web".
3. It might require additional API calls. Example: figuring out the path(s) associated with a specific file.

Usage

```
drive_reveal(
  file,
  what = c("path", "trashed", "mime_type", "permissions", "published")
)
```

Arguments

`file`	Something that identifies the file(s) of interest on your Google Drive. Can be a character vector of names/paths, a character vector of file ids or URLs marked with `as_id()`, or a `dribble`.
`what`	Character, describing the type of info you want to add: • path. Warning: this can be slow, especially if called on many files. • trashed • mime_type • permissions. Who is this file shared with and in which roles? • published

Value

An object of class dribble, a tibble with one row per item.

Special considerations for paths

Note that Google Drive does NOT behave like your local file system:

- File and folder names need not be unique, even at a given level of the hierarchy. A single name or file path can be associated with multiple files (or zero or exactly one).
- A file can have more than one direct parent. This implies that a single file can be represented by multiple paths.

Bottom line: Do not assume there is a one-to-one relationship between file name or path and a Drive file or folder. This implies the length of the input (i.e. the number of input paths or the number of rows in a dribble) will not necessarily equal the number rows in the output.

Trashed

When what = "trashed", the dribble gains a logical variable that indicates whether a file is in the trash.

MIME type

When what = "mime_type", the dribble gains a variable of MIME types.

Permissions

When what = "permissions" the dribble gains a logical variable shared that indicates whether a file is shared and a new list-column permissions_resource containing lists of Permissions resources.

Publishing

When what = "published" the dribble gains a logical variable published that indicates whether a file is published and a new list-column revision_resource containing lists of Revisions resources.

Examples

```
## Not run:
## Get a nice, random selection of files
files <- drive_find(n_max = 10, trashed = NA)

## Reveal
##   * paths (warning: can be slow for many files!)
##   * if `trashed` or not
##   * MIME type
##   * permissions, i.e. sharing status
##   * if `published` or not
drive_reveal(files, "path")
drive_reveal(files, "trashed")
drive_reveal(files, "mime_type")
drive_reveal(files, "permissions")
drive_reveal(files, "published")

## 'root' is a special file id that always represents your root folder
drive_get(id = "root") %>% drive_reveal("path")

## End(Not run)
```

`drive_rm` *Delete files from Drive*

Description

Caution: this will permanently delete your files! For a safer, reversible option, see `drive_trash()`.

Usage

```
drive_rm(..., verbose = TRUE)
```

Arguments

`...`	One or more Drive files, specified in any valid way, i.e. as a `dribble`, by name or path, or by file id or URL marked with `as_id()`. Or any combination thereof. Elements are processed with `as_dribble()` and row-bound prior to deletion.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

Logical vector, indicating whether the delete succeeded.

See Also

Wraps the `files.delete` endpoint:

- `https://developers.google.com/drive/v3/reference/files/delete`

Examples

```
## Not run:
## Create something to remove
drive_upload(drive_example("chicken.txt"), name = "chicken-rm.txt")

## Remove it by name
drive_rm("chicken-rm.txt")

## Create several things to remove
x1 <- drive_upload(drive_example("chicken.txt"), name = "chicken-abc.txt")
drive_upload(drive_example("chicken.txt"), name = "chicken-def.txt")
```

```
x2 <- drive_upload(drive_example("chicken.txt"), name = "chicken-ghi.txt")

## Remove them all at once, specified in different ways
drive_rm(x1, "chicken-def.txt", as_id(x2))

## End(Not run)
```

---

drive_share *Share Drive files*

Description

Grant individuals or other groups access to files, including permission to read, comment, or edit. The returned `dribble` will have extra columns, `shared` and `permissions_resource`. Read more in `drive_reveal()`.

`drive_share_anyone()` is a convenience wrapper for a common special case: "make this `file` readable by 'anyone with a link'".

Usage

```
drive_share(
  file,
  role = c("reader", "commenter", "writer", "fileOrganizer", "owner", "organizer"),
  type = c("user", "group", "domain", "anyone"),
  ...,
  verbose = TRUE
)

drive_share_anyone(file, verbose = TRUE)
```

Arguments

`file` Something that identifies the file(s) of interest on your Google Drive. Can be a character vector of names/paths, a character vector of file ids or URLs marked with `as_id()`, or a `dribble`.

`role` Character. The role to grant. Must be one of:

- organizer (applies only to Team Drives)
- owner
- fileOrganizer
- writer
- commenter
- reader

`type` Character. Describes the grantee. Must be one of:

- user
- group

- domain
- anyone

`...`	Name-value pairs to add to the API request. This is where you provide additional information, such as the `emailAddress` (when grantee `type` is `"group"` or `"user"`) or the `domain` (when grantee type is `"domain"`). Read the API docs linked below for more details.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `permissions.create` endpoint:

- `https://developers.google.com/drive/v3/reference/permissions/create`

Drive roles and permissions are described here:

- `https://developers.google.com/drive/api/v3/ref-roles`

Examples

```
## Not run:
## Upload a file to share
file <- drive_upload(
   drive_example("chicken.txt"),
   name = "chicken-share.txt",
   type = "document"
)

## Let a specific person comment
file <- file %>%
  drive_share(
    role = "commenter",
    type = "user",
    emailAddress = "susan@example.com"
)

## Let a different specific person edit and customize the email notification
file <- file %>%
  drive_share(
    role = "writer",
    type = "user",
    emailAddress = "carol@example.com",
    emailMessage = "Would appreciate your feedback on this!"
)

## Let anyone read the file
file <- file %>%
  drive_share(role = "reader", type = "anyone")
```

```
## Single-purpose wrapper function for this
drive_share_anyone(file)

## Clean up
drive_rm(file)

## End(Not run)
```

drive_token *Produce configured token*

Description

For internal use or for those programming around the Drive API. Returns a token pre-processed with `httr::config()`. Most users do not need to handle tokens "by hand" or, even if they need some control, `drive_auth()` is what they need. If there is no current token, `drive_auth()` is called to either load from cache or initiate OAuth2.0 flow. If auth has been deactivated via `drive_deauth()`, `drive_token()` returns `NULL`.

Usage

```
drive_token()
```

Value

A `request` object (an S3 class provided by httr).

See Also

Other low-level API functions: `drive_has_token()`, `request_generate()`, `request_make()`

Examples

```
## Not run:
req <- request_generate(
  "drive.files.get",
  list(fileId = "abc"),
  token = drive_token()
)
req

## End(Not run)
```

drive_trash — *Move Drive files to or from trash*

Description

Move Drive files to or from trash

Usage

```
drive_trash(file, verbose = TRUE)

drive_untrash(file, verbose = TRUE)
```

Arguments

`file`	Something that identifies the file(s) of interest on your Google Drive. Can be a character vector of names/paths, a character vector of file ids or URLs marked with `as_id()`, or a `dribble`.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
## Create a file and put it in the trash.
file <- drive_upload(drive_example("chicken.txt"), "chicken-trash.txt")
drive_trash("chicken-trash.txt")

## Confirm it's in the trash
drive_find(trashed = TRUE)

## Remove it from the trash and confirm
drive_untrash("chicken-trash.txt")
drive_find(trashed = TRUE)

## Clean up
drive_rm("chicken-trash.txt")

## End(Not run)
```

`drive_update` *Update an existing Drive file*

Description

Update an existing Drive file id with new content ("media" in Drive API-speak), new metadata, or both. To create a new file or update existing, depending on whether the Drive file already exists, see `drive_put()`.

Usage

```
drive_update(file, media = NULL, ..., verbose = TRUE)
```

Arguments

`file`	Something that identifies the file of interest on your Google Drive. Can be a name or path, a file id or URL marked with `as_id()`, or a `dribble`.
`media`	Character, path to the local file to upload.
`...`	Named parameters to pass along to the Drive API. Has the tidy dots semantics that come from using `rlang::list2()`. You can affect the metadata of the target file by specifying properties of the Files resource via .... Read the "Request body" section of the Drive API docs for the associated endpoint to learn about relevant parameters.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `files.update` endpoint:

- `https://developers.google.com/drive/v3/reference/files/update`

This function supports media upload:

- `https://developers.google.com/drive/v3/web/manage-uploads`

Examples

```
## Not run:
## Create a new file, so we can update it
x <- drive_upload(drive_example("chicken.csv"))

## Update the file with new media
x <- x %>%
  drive_update(drive_example("chicken.txt"))
```

```
## Update the file with new metadata.
## Notice here `name` is not an argument of `drive_update()`, we are passing
## this to the API via the `...``
x <- x %>%
  drive_update(name = "CHICKENS!")

## We can add a parent folder by passing `addParents` via `...`.
folder <- drive_mkdir("second-parent-folder")
x <- x %>%
  drive_update(addParents = as_id(folder))
## Verify the file now has multiple parents
purrr::pluck(x, "drive_resource", 1, "parents")

## Update the file with new media AND new metadata
x <- x %>%
  drive_update(drive_example("chicken.txt"), name = "chicken-poem-again.txt")

## Clean up
drive_rm(x, folder)

## End(Not run)
```

`drive_upload` *Upload into a new Drive file*

Description

Uploads a local file into a new Drive file. To update the content or metadata of an existing Drive file, use `drive_update()`. To upload or update, depending on whether the Drive file already exists, see `drive_put()`.

Usage

```
drive_upload(
  media,
  path = NULL,
  name = NULL,
  type = NULL,
  ...,
  overwrite = NA,
  verbose = TRUE
)
```

Arguments

`media`	Character, path to the local file to upload.

`path`	Specifies target destination for the new file on Google Drive. Can be an actual path (character), a file id marked with `as_id()`, or a `dribble`. If specified as an actual path, it is best to explicitly indicate if it's a folder by including a trailing slash, since it cannot always be worked out from the context of the call. Will default to its local name.
`name`	Character, new file name if not specified as part of `path`. This will force `path` to be treated as a folder, even if it is character and lacks a trailing slash. Will default to its local name.
`type`	Character. If `type = NULL`, a MIME type is automatically determined from the file extension, if possible. If the source file is of a suitable type, you can request conversion to Google Doc, Sheet or Slides by setting `type` to `document`, `spreadsheet`, or `presentation`, respectively. All non-`NULL` values for `type` are pre-processed with `drive_mime_type()`.
`...`	Named parameters to pass along to the Drive API. Has the tidy dots semantics that come from using `rlang::list2()`. You can affect the metadata of the target file by specifying properties of the Files resource via `...`. Read the "Request body" section of the Drive API docs for the associated endpoint to learn about relevant parameters.
`overwrite`	Logical, indicating whether to check for a pre-existing file at the targetted "filepath". The quotes around "filepath" refer to the fact that Drive does not impose a 1-to-1 relationship between filepaths and files, like a typical file system; read more about that in `drive_get()`. • `NA` (default): Just do the operation, even if it results in multiple files with the same filepath. • `TRUE`: Check for a pre-existing file at the filepath. If there is zero or one, move a pre-existing file to the trash, then carry on. Note that the new file does not inherit any properties from the old one, such as sharing or publishing settings. It will have a new file ID. An error is thrown if two or more pre-existing files are found. • `FALSE`: Error if there is any pre-existing file at the filepath. Note that existence checks, based on filepath, are expensive operations, i.e. they require additional API calls.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `files.create` endpoint:

- `https://developers.google.com/drive/v3/reference/files/create`

MIME types that can be converted to native Google formats:

- `https://developers.google.com/drive/v3/web/manage-uploads#importing_to_google_docs_types_wzxhzdk18wzxhzdk19`

Examples

```
## Not run:
## upload a csv file
chicken_csv <- drive_upload(
  drive_example("chicken.csv"),
  "chicken-upload.csv"
)

## or convert it to a Google Sheet
chicken_sheet <- drive_upload(
  drive_example("chicken.csv"),
  name = "chicken-sheet-upload.csv",
  type = "spreadsheet"
)

## check out the new Sheet!
drive_browse(chicken_sheet)

## clean-up
drive_find("chicken.*upload") %>% drive_rm()

## Upload a file and, at the same time, star it
chicken <- drive_upload(
  drive_example("chicken.jpg"),
  starred = "true"
)

## Is is really starred? YES
purrr::pluck(chicken, "drive_resource", 1, "starred")

## Clean up
drive_rm(chicken)

## `overwrite = FALSE` errors if something already exists at target filepath
## THIS WILL ERROR!
drive_create("name-squatter")
drive_upload(
  drive_example("chicken.jpg"),
  name = "name-squatter",
  overwrite = FALSE
)

## `overwrite = TRUE` moves the existing item to trash, then proceeds
chicken <- drive_upload(
  drive_example("chicken.jpg"),
  name = "name-squatter",
  overwrite = TRUE
)

## Clean up
drive_rm(chicken)
```

```
## Upload to a Team Drive:
##   * your Google account must have Team Drive privileges, obviously
##   * the Team Drive (or Team Drive-hosted folder) MUST be captured as a
##     dribble first and provided via `path`
td <- team_drive_get("Marketing")
drive_upload("fascinating.csv", path = td)

## End(Not run)
```

drive_user *Get info on current user*

Description

Reveals information about the user associated with the current token. This is a thin wrapper around `drive_about()` that just extracts the most useful information (the information on current user) and prints it nicely.

Usage

```
drive_user(verbose = TRUE)
```

Arguments

verbose	Logical, indicating whether to print informative messages (default TRUE).

Value

A list of class drive_user.

See Also

Wraps the about.get endpoint:

- https://developers.google.com/drive/v3/reference/about/get

Examples

```
## Not run:
drive_user()

## more info is returned than is printed
user <- drive_user()
user[["permissionId"]]

## End(Not run)
```

request_generate *Build a request for the Google Drive API*

Description

Build a request, using knowledge of the Drive v3 API from its Discovery Document. Most users should, instead, use higher-level wrappers that facilitate common tasks, such as uploading or downloading Drive files. The functions here are intended for internal use and for programming around the Drive API.

`request_generate()` lets you provide the bare minimum of input. It takes a nickname for an endpoint and:

- Uses the API spec to look up the `path`, `method`, and base URL.
- Checks `params` for validity and completeness with respect to the endpoint. Separates parameters into those destined for the body, the query, and URL endpoint substitution (which is also enacted).
- Adds an API key to the query if and only if `token` = `NULL`.
- Adds `supportsTeamDrives` = `TRUE` to the query if the endpoint requires.

Usage

```
request_generate(
  endpoint = character(),
  params = list(),
  key = NULL,
  token = drive_token()
)
```

Arguments

`endpoint`	Character. Nickname for one of the selected Drive v3 API endpoints built into googledrive. Learn more in `drive_endpoints()`.
`params`	Named list. Parameters destined for endpoint URL substitution, the query, or the body.
`key`	API key. Needed for requests that don't contain a token. The need for an API key in the absence of a token is explained in Google's document Credentials, access, security, and identity. In order of precedence, these sources are consulted: the formal `key` argument, a `key` parameter in `params`, a user-configured API key fetched via `drive_api_key()`, a built-in key shipped with googledrive. See `drive_auth_configure()` for details on a user-configured key.
`token`	Drive token. Set to `NULL` to suppress the inclusion of a token. Note that, if auth has been de-activated via `drive_deauth()`, `drive_token()` will actually return `NULL`.

Value

`list()`
Components are `method`, `path`, `query`, `body`, `token`, and `url`, suitable as input for `request_make()`.

See Also

`gargle::request_develop()`, `gargle::request_build()`

Other low-level API functions: `drive_has_token()`, `drive_token()`, `request_make()`

Examples

```
## Not run:
req <- request_generate(
  "drive.files.get",
  list(fileId = "abc"),
  token = drive_token()
)
req

## End(Not run)
```

`request_make` *Make a request for the Google Drive v3 API*

Description

Low-level functions to execute one or more Drive API requests and, perhaps, process the response(s). Most users should, instead, use higher-level wrappers that facilitate common tasks, such as uploading or downloading Drive files. The functions here are intended for internal use and for programming around the Drive API. Three functions are documented here:

- `request_make()` does the bare minimum: calls `gargle::request_make()`, only adding the googledrive user agent. Typically the input is created with `request_generate()` and the output is processed with `gargle::response_process()`.
- `do_request()` is simply `gargle::response_process(request_make(x, ...))`. It exists only because we had to make `do_paginated_request()` and it felt weird to not make the equivalent for a single request.
- `do_paginated_request()` executes the input request **with page traversal**. It is impossible to separate paginated requests into a "make request" step and a "process request" step, because the token for the next page must be extracted from the content of the current page. Therefore this function does both and returns a list of processed responses, one per page.

Usage

```
request_make(x, ...)

do_request(x, ...)

do_paginated_request(x, ..., n_max = Inf, n = function(res) 1, verbose = TRUE)
```

Arguments

`x`	List, holding the components for an HTTP request, presumably created with `request_generate()` Should contain the `method`, `url`, `body`, and `token`.
`...`	Optional arguments passed through to the HTTP method.
`n_max`	Maximum number of items to return. Defaults to `Inf`, i.e. there is no limit and we keep making requests until we get all items.
`n`	Function that computes the number of items in one response or page. The default function always returns 1 and therefore treats each page as an item. If you know more about the structure of the response, you can pass another function to count and threshhold, for example, the number of files or comments.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

`request_make()`: Object of class `response` from httr.

`do_request()`: List representing the content returned by a single request.

`do_paginated_request()`: List of lists, representing the returned content, one component per page.

See Also

Other low-level API functions: `drive_has_token()`, `drive_token()`, `request_generate()`

Examples

```
## Not run:
## build a request for an endpoint that is:
##   * paginated
##   * NOT privileged in googledrive, i.e. not covered by request_generate()
## "comments" are a great example
## https://developers.google.com/drive/v3/reference/comments
##
## Practice with a target file with > 2 comments
## Note that we request 2 items (comments) per page
req <- build_request(
  path = "drive/v3/files/{fileId}/comments",
  method = "GET",
  params = list(
    fileId = "your-file-id-goes-here",
    fields = "*",
    pageSize = 2
  ),
  token = googledrive::drive_token()
)
## make the paginated request, but cap it at 1 page
## should get back exactly two comments
do_paginated_request(req, n_max = 1)

## End(Not run)
```

team_drives *Access Team Drives*

Description

How to capture a Team Drive or files/folders that live on a Team Drive for downstream use:

- `drive_find()` and `drive_get()` return a `dribble` with metadata on files, including folders. Both can be directed to search on Team Drives, using the optional arguments `team_drive` or `corpus` (documented below).
- `team_drive_find()` and `team_drive_get()` return a `dribble` with metadata on Team Drives themselves. You will need this in order to use a Team Drive in certain file operations. For example, you can specify a Team Drive as the parent folder via the `path` argument for upload, move, copy, etc.

Regard the functions above as the official "port of entry" for Team Drives and Team Drive files and folders. Use them to capture your target(s) in a `dribble` to pass along to other googledrive functions. The general flexibility to refer to files by name, path, or id does not apply to Team Drive files. While it's always a good idea to get things into a `dribble` early, for Team Drives it's absolutely required.

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Specific Team Drive

To search one specific Team Drive, pass its name, marked id, or `dribble` to `team_drive` somewhere in the call, like so:

```
drive_find(..., team_drive = "i_am_a_team_drive_name")
drive_find(..., team_drive = as_id("i_am_a_team_drive_id"))
drive_find(..., team_drive = i_am_a_team_drive_dribble)
```

The value of `team_drive` is pre-processed with `as_team_drive()`.

Other collections

To search other collections, pass the `corpus` parameter somewhere in the call, like so:

```
drive_find(..., corpus = "user")
drive_find(..., corpus = "all")
drive_find(..., corpus = "domain")
```

Possible values of `corpus` and what they mean:

- `"user"`: Queries files that the user has accessed, including both Team and non-Team Drive files.

- `"all"`: Queries files that the user has accessed and all Team Drives in which they are a member. If you're reading the Drive API docs, this is a googledrive convenience short cut for `"user,allTeamDrives"`.
- `"domain"`: Queries files that are shared to the domain, including both Team Drive and non-Team Drive files.

API docs

googledrive implements Team Drive support as outlined here:

- `https://developers.google.com/drive/v3/web/enable-teamdrives#including_team_drive_content_fileslist`

Users shouldn't need to know any of this, but here are details for the curious. The extra information needed to search Team Drives consists of the following query parameters:

- `corpora`: Where to search? Formed from googledrive's `corpus` argument.
- `teamDriveId`: The id of a specific Team Drive. Only allowed – and also absolutely required – when `corpora = "teamDrive"`. When user specifies a Team Drive, googledrive sends its id and also infers that `corpora` should be set to `"teamDrive"` and sent.
- `includeTeamDriveItems`: Do you want to see Team Drive items? Obviously, this should be `TRUE` and googledrive sends this whenever Team Drive parameters are detected
- `supportsTeamDrives`: Does the sending application (googledrive, in this case) know about Team Drives? Obviously, this should be `TRUE` and googledrive sends it for all applicable endpoints, all the time.

`team_drive_create` *Create a new Team Drive*

Description

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
team_drive_create(name, verbose = TRUE)
```

Arguments

`name`	Character. Name of the new Team Drive. Must be non-empty and not entirely whitespace.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `teamdrives.create` endpoint:

- https://developers.google.com/drive/v3/reference/teamdrives/create

Examples

```
## Not run:
team_drive_create("my-awesome-team-drive")

## clean up
team_drive_rm("my-awesome-team-drive")

## End(Not run)
```

`team_drive_find` *Find Team Drives*

Description

This is the closest googledrive function to what you get from visiting https://drive.google.com and clicking "Team Drives".

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
team_drive_find(pattern = NULL, n_max = Inf, ..., verbose = TRUE)
```

Arguments

`pattern`	Character. If provided, only the items whose names match this regular expression are returned. This is implemented locally on the results returned by the API.
`n_max`	Integer. An upper bound on the number of items to return. This applies to the results requested from the API, which may be further filtered locally, via the `pattern` argument.
`...`	Other parameters to pass along in the request, such as `pageSize`.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `teamdrives.list` endpoint::

- https://developers.google.com/drive/v3/reference/teamdrives/list

Examples

```
## Not run:
team_drive_find()

## End(Not run)
```

`team_drive_get` *Get Team Drives by name or id*

Description

Retrieve metadata for Team Drives specified via name or id. Note that Google Drive does NOT behave like your local file system:

- You can get zero, one, or more Team Drives back for each name! Team Drive names need not be unique.

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
team_drive_get(name = NULL, id = NULL, verbose = TRUE)
```

Arguments

`name`	Character vector of names. A character vector marked with `as_id()` is treated as if it was provided via the `id` argument.
`id`	Character vector of Team Drive ids or URLs (it is first processed with `as_id()`). If both `name` and `id` are non-`NULL`, `id` is silently ignored.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

Examples

```
## Not run:
team_drive_get("my-awesome-team-drive")
team_drive_get(c("apple", "orange", "banana"))
team_drive_get(as_id("KCmiHLXUk9PVA-0AJNG"))
team_drive_get(as_id("https://drive.google.com/drive/u/0/folders/KCmiHLXUk9PVA-0AJNG"))
team_drive_get(id = "KCmiHLXUk9PVA-0AJNG")
team_drive_get(id = "https://drive.google.com/drive/u/0/folders/KCmiHLXUk9PVA-0AJNG")

## End(Not run)
```

team_drive_rm *Delete Team Drives*

Description

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
team_drive_rm(team_drive = NULL, verbose = TRUE)
```

Arguments

`team_drive`	Anything that identifies the Team Drive(s) of interest. Can be a character vector of names, a character vector of file ids or URLs marked with `as_id()`, or a `dribble` consisting only of Team Drives.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

Logical vector, indicating whether the delete succeeded.

See Also

Wraps the `teamdrives.delete` endpoint:

- https://developers.google.com/drive/v3/reference/teamdrives/delete

Examples

```
## Not run:
## Create Team Drives to remove in various ways
team_drive_create("testdrive-01")
td02 <- team_drive_create("testdrive-02")
team_drive_create("testdrive-03")
td04 <- team_drive_create("testdrive-04")

## remove by name
team_drive_rm("testdrive-01")
## remove by id
team_drive_rm(as_id(td02))
## remove by URL (or, rather, id found in URL)
team_drive_rm(as_id("https://drive.google.com/drive/u/0/folders/Q5DqUk9PVA"))
## remove by dribble
team_drive_rm(td04)

## End(Not run)
```

`team_drive_update` *Update an existing Team Drive*

Description

Update the metadata of an existing Team Drive, e.g. its background image or theme.

Note: Team Drives are only available to users of certain enhanced Google services, such as G Suite Enterprise, G Suite Business, or G Suite for Education.

Usage

```
team_drive_update(team_drive, ..., verbose = TRUE)
```

Arguments

`team_drive`	Anything that identifies one specific Team Drive: its name, its id or URL marked with `as_id()`, or a `dribble`. Is pre-processed with `as_team_drive()`. Read more about Team Drives.
`...`	Named parameters to pass along to the Drive API. See the "Request body" section of the Drive API docs for the associated endpoint.
`verbose`	Logical, indicating whether to print informative messages (default `TRUE`).

Value

An object of class `dribble`, a tibble with one row per item.

See Also

Wraps the `teamdrives.update` endpoint:

- `https://developers.google.com/drive/v3/reference/teamdrives/update`

Examples

```
## Not run:
## create a Team Drive
td <- team_drive_create("I love themes!")

## see the themes available to you
themes <- drive_user(fields = "teamDriveThemes")$teamDriveThemes
purrr::map_chr(themes, "id")

## cycle through various themes for this Team Drive
td <- team_drive_update(td, themeId = "bok_choy")
td <- team_drive_update(td, themeId = "cocktails")

## clean up
team_drive_rm(td)
```

```
## End(Not run)
```

Index