Google Drive

Paths are specified as `drive:path`

Drive paths may be as deep as required, e.g. `drive:directory/subdirectory`.

The initial setup for drive involves getting a token from Google drive which you need to do in your browser. `rclone config` walks you through it.

Here is an example of how to make a remote called `remote`. First run:

```
rclone config
```

This will guide you through an interactive setup process:

```
No remotes found - make a new one
n) New remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
n/r/c/s/q> n
name> remote
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Google Drive
   \ "drive"
[snip]
Storage> drive
Google Application Client Id - leave blank normally.
client_id>
Google Application Client Secret - leave blank normally.
client_secret>
Scope that rclone should use when requesting access from drive.
Choose a number from below, or type in your own value
 1 / Full access all files, excluding Application Data Folder.
   \ "drive"
 2 / Read-only access to file metadata and file contents.
   \ "drive.readonly"
   / Access to files created by rclone only.
 3 | These are visible in the drive website.
   | File authorization is revoked when the user deauthorizes the app.
   \ "drive.file"
   / Allows read and write access to the Application Data folder.
 4 | This is not visible in the drive website.
   \ "drive.appfolder"
   / Allows read-only access to file metadata but
 5 | does not allow any access to read or download file content.
   \ "drive.metadata.readonly"
scope> 1
ID of the root folder - leave blank normally.  Fill in to access "Computers" folders. (see do
root_folder_id>
Service Account Credentials JSON file path - needed only if you want use SA instead of intera
service_account_file>
Remote config
Use auto config?
 * Say Y if not sure
 * Say N if you are working on a remote or headless machine or Y didn't work
y) Yes
n) No
y/n> y
If your browser doesn't open automatically go to the following link: http://127.0.0.1:53682/a
Log in and authorize rclone for access
Waiting for code...
Got code
```

```
Configure this as a team drive?
y) Yes
n) No
y/n> n
--------------------
[remote]
client_id =
client_secret =
scope = drive
root_folder_id =
service_account_file =
token = {"access_token":"XXX","token_type":"Bearer","refresh_token":"XXX","expiry":"2014-03-1
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

Note that rclone runs a webserver on your local machine to collect the token as returned from Google if you use auto config mode. This only runs from the moment it opens your browser to the moment you get back the verification code. This is on `http://127.0.0.1:53682/` and this it may require you to unblock it temporarily if you are running a host firewall, or use manual mode.

You can then use it like this,

List directories in top level of your drive

```
rclone lsd remote:
```

List all the files in your drive

```
rclone ls remote:
```

To copy a local directory to a drive directory called backup

```
rclone copy /home/source remote:backup
```

Scopes

Rclone allows you to select which scope you would like for rclone to use. This changes what type of token is granted to rclone. [The scopes are defined here](#).

The scope are

drive

This is the default scope and allows full access to all files, except for the Application Data Folder (see below).

Choose this one if you aren't sure.

drive.readonly

This allows read only access to all files. Files may be listed and downloaded but not uploaded, renamed or deleted.

drive.file

With this scope rclone can read/view/modify only those files and folders it creates.

So if you uploaded files to drive via the web interface (or any other means) they will not be visible to rclone.

This can be useful if you are using rclone to backup data and you want to be sure confidential data on your drive is not visible to rclone.

Files created with this scope are visible in the web interface.

drive.appfolder

This gives rclone its own private area to store files. Rclone will not be able to see any other files on your drive and you won't be able to see rclone's files from the web interface either.

drive.metadata.readonly

This allows read only access to file names only. It does not allow rclone to download or upload data, or rename or delete files or directories.

Root folder ID

You can set the `root_folder_id` for rclone. This is the directory (identified by its `Folder ID`) that rclone considers to be the root of your drive.

Normally you will leave this blank and rclone will determine the correct root to use itself.

However you can set this to restrict rclone to a specific folder hierarchy or to access data within the "Computers" tab on the drive web interface (where files from Google's Backup and Sync desktop program go).

In order to do this you will have to find the `Folder ID` of the directory you wish rclone to display. This will be the last segment of the URL when you open the relevant folder in the drive web interface.

So if the folder you want rclone to use has a URL which looks like`https://drive.google.com/drive/folders/1XyfxxxxxxxxxxxxxxxxxxxxxxxxxKHCh` in the browser, then you use `1XyfxxxxxxxxxxxxxxxxxxxxxxxxxKHCh` as the `root_folder_id` in the config.

NB folders under the "Computers" tab seem to be read only (drive gives a 500 error) when using rclone.

There doesn't appear to be an API to discover the folder IDs of the "Computers" tab - please contact us if you know otherwise!

Note also that rclone can't access any data under the "Backups" tab on the google drive web interface yet.

Service Account support

You can set up rclone with Google Drive in an unattended mode, i.e. not tied to a specific end-user Google account. This is useful when you want to synchronise files onto machines that don't have actively logged-in users, for example build machines.

To use a Service Account instead of OAuth2 token flow, enter the path to your Service Account credentials at the `service_account_file`prompt during `rclone config` and rclone won't use the browser based authentication flow. If you'd rather stuff the contents of the credentials file into the rclone config file, you can set `service_account_credentials` with the actual contents of the file instead, or set the equivalent environment variable.

Use case - Google Apps/G-suite account and individual Drive

Let's say that you are the administrator of a Google Apps (old) or G-suite account. The goal is to store data on an individual's Drive account, who IS a member of the domain. We'll call the domain **example.com**, and the user **foo@example.com**.

There's a few steps we need to go through to accomplish this:

1. Create a service account for example.com

- To create a service account and obtain its credentials, go to the Google Developer Console.
- You must have a project - create one if you don't.
- Then go to "IAM & admin" -> "Service Accounts".
- Use the "Create Credentials" button. Fill in "Service account name" with something that identifies your client. "Role" can be empty.
- Tick "Furnish a new private key" - select "Key type JSON".
- Tick "Enable G Suite Domain-wide Delegation". This option makes "impersonation" possible, as documented here: Delegating domain-wide authority to the service account
- These credentials are what rclone will use for authentication. If you ever need to remove access, press the "Delete service account key" button.

2. Allowing API access to example.com Google Drive

- Go to example.com's admin console
- Go into "Security" (or use the search bar)
- Select "Show more" and then "Advanced settings"
- Select "Manage API client access" in the "Authentication" section
- In the "Client Name" field enter the service account's "Client ID" - this can be found in the Developer Console under "IAM & Admin" -> "Service Accounts", then "View Client ID" for the newly created service account. It is a ~21 character numerical string.
- In the next field, "One or More API Scopes", enter `https://www.googleapis.com/auth/drive` to grant access to Google Drive specifically.

3. Configure rclone, assuming a new install

```
rclone config

n/s/q> n         # New
name>gdrive      # Gdrive is an example name
Storage>         # Select the number shown for Google Drive
client_id>       # Can be left blank
client_secret>   # Can be left blank
scope>           # Select your scope, 1 for example
root_folder_id>  # Can be left blank
service_account_file> /home/foo/myJSONfile.json # This is where the JSON file goes!
y/n>             # Auto config, y
```

4. Verify that it's working

- `rclone -v --drive-impersonate foo@example.com lsf gdrive:backup`
- The arguments do:
  - `-v` - verbose logging
  - `--drive-impersonate foo@example.com` - this is what does the magic, pretending to be user foo.
  - `lsf` - list files in a parsing friendly way
  - `gdrive:backup` - use the remote called gdrive, work in the folder named backup.

Note: in case you configured a specific root folder on gdrive and rclone is unable to access the contents of that folder when using `--drive-impersonate`, do this instead:

- in the gdrive web interface, share your root folder with the user/email of the new Service Account you created/selected at step #1
- use rclone without specifying the `--drive-impersonate` option, like this: `rclone -v foo@example.com lsf gdrive:backup`

Team drives

If you want to configure the remote to point to a Google Team Drive then answer `y` to the question `Configure this as a team drive?`.

This will fetch the list of Team Drives from google and allow you to configure which one you want to use. You can also type in a team drive ID if you prefer.

For example:

```
Configure this as a team drive?
y) Yes
n) No
y/n> y
Fetching team drive list...
Choose a number from below, or type in your own value
 1 / Rclone Test
   \ "xxxxxxxxxxxxxxxxxxxx"
 2 / Rclone Test 2
   \ "yyyyyyyyyyyyyyyyyyyy"
 3 / Rclone Test 3
   \ "zzzzzzzzzzzzzzzzzzzz"
Enter a Team Drive ID> 1
--------------------
[remote]
client_id =
client_secret =
token = {"AccessToken":"xxxx.x.xxxxx_xxxxxxxxxxx_xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
team_drive = xxxxxxxxxxxxxxxxxxxx
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

--fast-list

This remote supports `--fast-list` which allows you to use fewer transactions in exchange for more memory. See the rclone docs for more details.

It does this by combining multiple `list` calls into a single API request.

This works by combining many `'%s' in parents` filters into one expression. To list the contents of directories a, b and c, the following requests will be send by the regular `List` function:

```
trashed=false and 'a' in parents
trashed=false and 'b' in parents
trashed=false and 'c' in parents
```

These can now be combined into a single request:

```
trashed=false and ('a' in parents or 'b' in parents or 'c' in parents)
```

The implementation of `ListR` will put up to 50 `parents` filters into one request. It will use the `--checkers` value to specify the number of requests to run in parallel.

In tests, these batch requests were up to 20x faster than the regular method. Running the following command against different sized folders gives:

```
rclone lsjson -vv -R --checkers=6 gdrive:folder
```

small folder (220 directories, 700 files):

- without `--fast-list`: 38s
- with `--fast-list`: 10s

large folder (10600 directories, 39000 files):

- without `--fast-list`: 22:05 min
- with `--fast-list`: 58s

Modified time

Google drive stores modification times accurate to 1 ms.

Restricted filename characters

Only Invalid UTF-8 bytes will be replaced, as they can't be used in JSON strings.

In contrast to other backends, `/` can also be used in names and `.` or `..` are valid names.

Revisions

Google drive stores revisions of files. When you upload a change to an existing file to google drive using rclone it will create a new revision of that file.

Revisions follow the standard google policy which at time of writing was

- They are deleted after 30 days or 100 revisions (whatever comes first).
- They do not count towards a user storage quota.

Deleting files

By default rclone will send all files to the trash when deleting files. If deleting them permanently is required then use the `--drive-use-trash=false` flag, or set the equivalent environment variable.

Shortcuts

In March 2020 Google introduced a new feature in Google Drive called drive shortcuts (API). These will (by September 2020) replace the ability for files or folders to be in multiple folders at once.

Shortcuts are files that link to other files on Google Drive somewhat like a symlink in unix, except they point to the underlying file data (e.g. the inode in unix terms) so they don't break if the source is renamed or moved about.

Be default rclone treats these as follows.

For shortcuts pointing to files:

- When listing a file shortcut appears as the destination file.
- When downloading the contents of the destination file is downloaded.
- When updating shortcut file with a non shortcut file, the shortcut is removed then a new file is uploaded in place of the shortcut.
- When server-side moving (renaming) the shortcut is renamed, not the destination file.
- When server-side copying the shortcut is copied, not the contents of the shortcut.
- When deleting the shortcut is deleted not the linked file.
- When setting the modification time, the modification time of the linked file will be set.

For shortcuts pointing to folders:

- When listing the shortcut appears as a folder and that folder will contain the contents of the linked folder appear (including any sub folders)
- When downloading the contents of the linked folder and sub contents are downloaded
- When uploading to a shortcut folder the file will be placed in the linked folder
- When server-side moving (renaming) the shortcut is renamed, not the destination folder
- When server-side copying the contents of the linked folder is copied, not the shortcut.
- When deleting with `rclone rmdir` or `rclone purge` the shortcut is deleted not the linked folder.
- **NB** When deleting with `rclone remove` or `rclone mount` the contents of the linked folder will be deleted.

The rclone backend command can be used to create shortcuts.

Shortcuts can be completely ignored with the `--drive-skip-shortcuts` flag or the corresponding `skip_shortcuts` configuration setting.

Emptying trash

If you wish to empty your trash you can use the `rclone cleanup remote:` command which will permanently delete all your trashed files. This command does not take any path arguments.

Note that Google Drive takes some time (minutes to days) to empty the trash even though the command returns within a few seconds. No output is echoed, so there will be no confirmation even using -v or -vv.

Quota information

To view your current quota you can use the `rclone about remote:` command which will display your usage limit (quota), the usage in Google Drive, the size of all files in the Trash and the space used by other Google services such as Gmail. This command does not take any path arguments.

Import/Export of google documents

Google documents can be exported from and uploaded to Google Drive.

When rclone downloads a Google doc it chooses a format to download depending upon the `--drive-export-formats` setting. By default the export formats are `docx,xlsx,pptx,svg` which are a sensible default for an editable document.

When choosing a format, rclone runs down the list provided in order and chooses the first file format the doc can be exported as from the list. If the file can't be exported to a format on the formats list, then rclone will choose a format from the default list.

If you prefer an archive copy then you might use `--drive-export-formats pdf`, or if you prefer openoffice/libreoffice formats you might use `--drive-export-formats ods,odt,odp`.

Note that rclone adds the extension to the google doc, so if it is called `My Spreadsheet` on google docs, it will be exported as `My Spreadsheet.xlsx` or `My Spreadsheet.pdf` etc.

When importing files into Google Drive, rclone will convert all files with an extension in `--drive-import-formats` to their associated document type. rclone will not convert any files by default, since the conversion is lossy process.

The conversion must result in a file with the same extension when the `--drive-export-formats` rules are applied to the uploaded document.

Here are some examples for allowed and prohibited conversions.

export-formats	import-formats	Upload Ext	Document Ext	Allowed
odt	odt	odt	odt	Yes
odt	docx,odt	odt	odt	Yes
	docx	docx	docx	Yes
	odt	odt	docx	No
odt,docx	docx,odt	docx	odt	No
docx,odt	docx,odt	docx	docx	Yes
docx,odt	docx,odt	odt	docx	No

This limitation can be disabled by specifying `--drive-allow-import-name-change`. When using this flag, rclone can convert multiple files types resulting in the same document type at once, e.g. with `--drive-import-formats docx,odt,txt`, all files having these extension would result in a document represented as a docx file. This brings the additional risk of overwriting a document, if multiple files have the same stem. Many rclone operations will not handle this name change in any way. They assume an equal name when copying files and might copy the file again or delete them when the name changes.

Here are the possible export extensions with their corresponding mime types. Most of these can also be used for importing, but there more that are not listed here. Some of these additional ones might only be available when the operating system provides the correct MIME type entries.

This list can be changed by Google Drive at any time and might not represent the currently available conversions.

Extension	Mime Type	Description
csv	text/csv	Standard CSV format for Spreadsheets
docx	application/vnd.openxmlformats-officedocument.wordprocessingml.document	Microsoft Office Document
epub	application/epub+zip	E-book format
html	text/html	An HTML Document
jpg	image/jpeg	A JPEG Image File
json	application/vnd.google-apps.script+json	JSON Text Format

Extension	Mime Type	Description
odp	application/vnd.oasis.opendocument.presentation	Openoffice Presentation
ods	application/vnd.oasis.opendocument.spreadsheet	Openoffice Spreadsheet
ods	application/x-vnd.oasis.opendocument.spreadsheet	Openoffice Spreadsheet
odt	application/vnd.oasis.opendocument.text	Openoffice Document
pdf	application/pdf	Adobe PDF Format
png	image/png	PNG Image Format
pptx	application/vnd.openxmlformats-officedocument.presentationml.presentation	Microsoft Office Powerpoint
rtf	application/rtf	Rich Text Format
svg	image/svg+xml	Scalable Vector Graphics Format
tsv	text/tab-separated-values	Standard TSV format for spreadsheets
txt	text/plain	Plain Text
xlsx	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	Microsoft Office Spreadsheet
zip	application/zip	A ZIP file of HTML, Images CSS

Google documents can also be exported as link files. These files will open a browser window for the Google Docs website of that document when opened. The link file extension has to be specified as a `--drive-export-formats` parameter. They will match all available Google Documents.

Extension	Description	OS Support
desktop	freedesktop.org specified desktop entry	Linux
link.html	An HTML Document with a redirect	All
url	INI style link file	macOS, Windows
webloc	macOS specific XML format	macOS

Standard Options

Here are the standard options specific to drive (Google Drive).

--drive-client-id

Google Application Client Id Setting your own is recommended. See https://rclone.org/drive/#making-your-own-client-id for how to create your own. If you leave this blank, it will use an internal key which is low performance.

- Config: client_id
- Env Var: RCLONE_DRIVE_CLIENT_ID
- Type: string
- Default: ""

--drive-client-secret

OAuth Client Secret Leave blank normally.

- Config: client_secret
- Env Var: RCLONE_DRIVE_CLIENT_SECRET
- Type: string
- Default: ""

--drive-scope

Scope that rclone should use when requesting access from drive.

- Config: scope
- Env Var: RCLONE_DRIVE_SCOPE
- Type: string
- Default: ""
- Examples:
    - "drive"
        - Full access all files, excluding Application Data Folder.
    - "drive.readonly"
        - Read-only access to file metadata and file contents.
    - "drive.file"
        - Access to files created by rclone only.
        - These are visible in the drive website.
        - File authorization is revoked when the user deauthorizes the app.
    - "drive.appfolder"
        - Allows read and write access to the Application Data folder.
        - This is not visible in the drive website.
    - "drive.metadata.readonly"
        - Allows read-only access to file metadata but
        - does not allow any access to read or download file content.

--drive-root-folder-id

ID of the root folder Leave blank normally.

Fill in to access "Computers" folders (see docs), or for rclone to use a non root folder as its starting point.

- Config: root_folder_id
- Env Var: RCLONE_DRIVE_ROOT_FOLDER_ID
- Type: string
- Default: ""

--drive-service-account-file

Service Account Credentials JSON file path Leave blank normally. Needed only if you want use SA instead of interactive login.

Leading `~` will be expanded in the file name as will environment variables such as `${RCLONE_CONFIG_DIR}`.

- Config: service_account_file
- Env Var: RCLONE_DRIVE_SERVICE_ACCOUNT_FILE
- Type: string
- Default: ""

--drive-alternate-export

Deprecated: no longer needed

- Config: alternate_export
- Env Var: RCLONE_DRIVE_ALTERNATE_EXPORT
- Type: bool
- Default: false

Advanced Options

Here are the advanced options specific to drive (Google Drive).

--drive-token

OAuth Access Token as a JSON blob.

- Config: token
- Env Var: RCLONE_DRIVE_TOKEN
- Type: string
- Default: ""

--drive-auth-url

Auth server URL. Leave blank to use the provider defaults.

- Config: auth_url
- Env Var: RCLONE_DRIVE_AUTH_URL
- Type: string
- Default: ""

--drive-token-url

Token server url. Leave blank to use the provider defaults.

- Config: token_url
- Env Var: RCLONE_DRIVE_TOKEN_URL
- Type: string
- Default: ""

--drive-service-account-credentials

Service Account Credentials JSON blob Leave blank normally. Needed only if you want use SA instead of interactive login.

- Config: service_account_credentials
- Env Var: RCLONE_DRIVE_SERVICE_ACCOUNT_CREDENTIALS
- Type: string
- Default: ""

--drive-team-drive

ID of the Team Drive

- Config: team_drive
- Env Var: RCLONE_DRIVE_TEAM_DRIVE
- Type: string
- Default: ""

--drive-auth-owner-only

Only consider files owned by the authenticated user.

- Config: auth_owner_only
- Env Var: RCLONE_DRIVE_AUTH_OWNER_ONLY
- Type: bool
- Default: false

--drive-use-trash

Send files to the trash instead of deleting permanently. Defaults to true, namely sending files to the trash. Use `--drive-use-trash=false`to delete files permanently instead.

- Config: use_trash
- Env Var: RCLONE_DRIVE_USE_TRASH
- Type: bool
- Default: true

--drive-skip-gdocs

Skip google documents in all listings. If given, gdocs practically become invisible to rclone.

- Config: skip_gdocs
- Env Var: RCLONE_DRIVE_SKIP_GDOCS
- Type: bool
- Default: false

--drive-skip-checksum-gphotos

Skip MD5 checksum on Google photos and videos only.

Use this if you get checksum errors when transferring Google photos or videos.

Setting this flag will cause Google photos and videos to return a blank MD5 checksum.

Google photos are identified by being in the "photos" space.

Corrupted checksums are caused by Google modifying the image/video but not updating the checksum.

- Config: skip_checksum_gphotos
- Env Var: RCLONE_DRIVE_SKIP_CHECKSUM_GPHOTOS
- Type: bool
- Default: false

--drive-shared-with-me

Only show files that are shared with me.

Instructs rclone to operate on your "Shared with me" folder (where Google Drive lets you access the files and folders others have shared with you).

This works both with the "list" (lsd, lsl, etc.) and the "copy" commands (copy, sync, etc.), and with all other commands too.

- Config: shared_with_me
- Env Var: RCLONE_DRIVE_SHARED_WITH_ME
- Type: bool
- Default: false

--drive-trashed-only

Only show files that are in the trash. This will show trashed files in their original directory structure.

- Config: trashed_only
- Env Var: RCLONE_DRIVE_TRASHED_ONLY
- Type: bool
- Default: false

--drive-starred-only

Only show files that are starred.

- Config: starred_only
- Env Var: RCLONE_DRIVE_STARRED_ONLY
- Type: bool
- Default: false

--drive-formats

Deprecated: see export_formats

- Config: formats
- Env Var: RCLONE_DRIVE_FORMATS
- Type: string
- Default: ""

--drive-export-formats

Comma separated list of preferred formats for downloading Google docs.

- Config: export_formats
- Env Var: RCLONE_DRIVE_EXPORT_FORMATS
- Type: string
- Default: "docx,xlsx,pptx,svg"

--drive-import-formats

Comma separated list of preferred formats for uploading Google docs.

- Config: import_formats
- Env Var: RCLONE_DRIVE_IMPORT_FORMATS
- Type: string
- Default: ""

--drive-allow-import-name-change

Allow the filetype to change when uploading Google docs (e.g. file.doc to file.docx). This will confuse sync and reupload every time.

- Config: allow_import_name_change
- Env Var: RCLONE_DRIVE_ALLOW_IMPORT_NAME_CHANGE
- Type: bool
- Default: false

--drive-use-created-date

Use file created date instead of modified date.,

Useful when downloading data and you want the creation date used in place of the last modified date.

WARNING: This flag may have some unexpected consequences.

When uploading to your drive all files will be overwritten unless they haven't been modified since their creation. And the inverse will occur while downloading. This side effect can be avoided by using the "--checksum" flag.

This feature was implemented to retain photos capture date as recorded by google photos. You will first need to check the "Create a Google Photos folder" option in your google drive settings. You can then copy or move the photos locally and use the date the image was taken (created) set as the modification date.

- Config: use_created_date

- Env Var: RCLONE_DRIVE_USE_CREATED_DATE
- Type: bool
- Default: false

--drive-use-shared-date

Use date file was shared instead of modified date.

Note that, as with "--drive-use-created-date", this flag may have unexpected consequences when uploading/downloading files.

If both this flag and "--drive-use-created-date" are set, the created date is used.

- Config: use_shared_date
- Env Var: RCLONE_DRIVE_USE_SHARED_DATE
- Type: bool
- Default: false

--drive-list-chunk

Size of listing chunk 100-1000. 0 to disable.

- Config: list_chunk
- Env Var: RCLONE_DRIVE_LIST_CHUNK
- Type: int
- Default: 1000

--drive-impersonate

Impersonate this user when using a service account.

- Config: impersonate
- Env Var: RCLONE_DRIVE_IMPERSONATE
- Type: string
- Default: ""

--drive-upload-cutoff

Cutoff for switching to chunked upload

- Config: upload_cutoff
- Env Var: RCLONE_DRIVE_UPLOAD_CUTOFF
- Type: SizeSuffix
- Default: 8M

--drive-chunk-size

Upload chunk size. Must a power of 2 >= 256k.

Making this larger will improve performance, but note that each chunk is buffered in memory one per transfer.

Reducing this will reduce memory usage but decrease performance.

- Config: chunk_size
- Env Var: RCLONE_DRIVE_CHUNK_SIZE
- Type: SizeSuffix
- Default: 8M

--drive-acknowledge-abuse

Set to allow files which return cannotDownloadAbusiveFile to be downloaded.

If downloading a file returns the error "This file has been identified as malware or spam and cannot be downloaded" with the error code "cannotDownloadAbusiveFile" then supply this flag to rclone to indicate you acknowledge the risks of downloading the file and rclone will download it anyway.

- Config: acknowledge_abuse
- Env Var: RCLONE_DRIVE_ACKNOWLEDGE_ABUSE
- Type: bool
- Default: false

--drive-keep-revision-forever

Keep new head revision of each file forever.

- Config: keep_revision_forever
- Env Var: RCLONE_DRIVE_KEEP_REVISION_FOREVER
- Type: bool
- Default: false

--drive-size-as-quota

Show sizes as storage quota usage, not actual size.

Show the size of a file as the storage quota used. This is the current version plus any older versions that have been set to keep forever.

WARNING: This flag may have some unexpected consequences.

It is not recommended to set this flag in your config - the recommended usage is using the flag form --drive-size-as-quota when doing rclone ls/lsl/lsf/lsjson/etc only.

If you do use this flag for syncing (not recommended) then you will need to use --ignore size also.

- Config: size_as_quota
- Env Var: RCLONE_DRIVE_SIZE_AS_QUOTA
- Type: bool
- Default: false

--drive-v2-download-min-size

If Object's are greater, use drive v2 API to download.

- Config: v2_download_min_size
- Env Var: RCLONE_DRIVE_V2_DOWNLOAD_MIN_SIZE
- Type: SizeSuffix
- Default: off

--drive-pacer-min-sleep

Minimum time to sleep between API calls.

- Config: pacer_min_sleep
- Env Var: RCLONE_DRIVE_PACER_MIN_SLEEP
- Type: Duration
- Default: 100ms

--drive-pacer-burst

Number of API calls to allow without sleeping.

- Config: pacer_burst
- Env Var: RCLONE_DRIVE_PACER_BURST
- Type: int
- Default: 100

--drive-server-side-across-configs

Allow server-side operations (e.g. copy) to work across different drive configs.

This can be useful if you wish to do a server-side copy between two different Google drives. Note that this isn't enabled by default because it isn't easy to tell if it will work between any two configurations.

- Config: server_side_across_configs
- Env Var: RCLONE_DRIVE_SERVER_SIDE_ACROSS_CONFIGS
- Type: bool
- Default: false

--drive-disable-http2

Disable drive using http2

There is currently an unsolved issue with the google drive backend and HTTP/2. HTTP/2 is therefore disabled by default for the drive backend but can be re-enabled here. When the issue is solved this flag will be removed.

See: https://github.com/rclone/rclone/issues/3631

- Config: disable_http2
- Env Var: RCLONE_DRIVE_DISABLE_HTTP2
- Type: bool
- Default: true

--drive-stop-on-upload-limit

Make upload limit errors be fatal

At the time of writing it is only possible to upload 750GB of data to Google Drive a day (this is an undocumented limit). When this limit is reached Google Drive produces a slightly different error message. When this flag is set it causes these errors to be fatal. These will stop the in-progress sync.

Note that this detection is relying on error message strings which Google don't document so it may break in the future.

See: https://github.com/rclone/rclone/issues/3857

- Config: stop_on_upload_limit
- Env Var: RCLONE_DRIVE_STOP_ON_UPLOAD_LIMIT
- Type: bool
- Default: false

--drive-stop-on-download-limit

Make download limit errors be fatal

At the time of writing it is only possible to download 10TB of data from Google Drive a day (this is an undocumented limit). When this limit is reached Google Drive produces a slightly different error message. When this flag is set it causes these errors to be fatal. These will stop the in-progress sync.

Note that this detection is relying on error message strings which Google don't document so it may break in the future.

- Config: stop_on_download_limit
- Env Var: RCLONE_DRIVE_STOP_ON_DOWNLOAD_LIMIT
- Type: bool
- Default: false

--drive-skip-shortcuts

If set skip shortcut files

Normally rclone dereferences shortcut files making them appear as if they are the original file (see the shortcuts section). If this flag is set then rclone will ignore shortcut files completely.

- Config: skip_shortcuts
- Env Var: RCLONE_DRIVE_SKIP_SHORTCUTS
- Type: bool
- Default: false

--drive-encoding

This sets the encoding for the backend.

See: the encoding section in the overview for more info.

- Config: encoding
- Env Var: RCLONE_DRIVE_ENCODING
- Type: MultiEncoder
- Default: InvalidUtf8

Backend commands

Here are the commands specific to the drive backend.

Run them with

```
rclone backend COMMAND remote:
```

The help below will explain what arguments each command takes.

See the "rclone backend" command for more info on how to pass options and arguments.

These can be run on a running backend using the rc command backend/command.

get

Get command for fetching the drive config parameters

```
rclone backend get remote: [options] [<arguments>+]
```

This is a get command which will be used to fetch the various drive config parameters

Usage Examples:

```
rclone backend get drive: [-o service_account_file] [-o chunk_size]
rclone rc backend/command command=get fs=drive: [-o service_account_file] [-o chunk_size]
```

Options:

- "chunk_size": show the current upload chunk size
- "service_account_file": show the current service account file

set

Set command for updating the drive config parameters

```
rclone backend set remote: [options] [<arguments>+]
```

This is a set command which will be used to update the various drive config parameters

Usage Examples:

```
rclone backend set drive: [-o service_account_file=sa.json] [-o chunk_size=67108864]
rclone rc backend/command command=set fs=drive: [-o service_account_file=sa.json] [-o chunk_s
```

Options:

- "chunk_size": update the current upload chunk size
- "service_account_file": update the current service account file

shortcut

Create shortcuts from files or directories

```
rclone backend shortcut remote: [options] [<arguments>+]
```

This command creates shortcuts from files or directories.

Usage:

```
rclone backend shortcut drive: source_item destination_shortcut
rclone backend shortcut drive: source_item -o target=drive2: destination_shortcut
```

In the first example this creates a shortcut from the "source_item" which can be a file or a directory to the "destination_shortcut". The "source_item" and the "destination_shortcut" should be relative paths from "drive:"

In the second example this creates a shortcut from the "source_item" relative to "drive:" to the "destination_shortcut" relative to "drive2:". This may fail with a permission error if the user authenticated with "drive2:" can't read files from "drive:".

Options:

- "target": optional target remote for the shortcut destination

drives

List the shared drives available to this account

```
rclone backend drives remote: [options] [<arguments>+]
```

This command lists the shared drives (teamdrives) available to this account.

Usage:

```
rclone backend drives drive:
```

This will return a JSON list of objects like this

```
[
    {
        "id": "0ABCDEF-01234567890",
        "kind": "drive#teamDrive",
        "name": "My Drive"
    },
    {
        "id": "0ABCDEFabcdefghijkl",
        "kind": "drive#teamDrive",
        "name": "Test Drive"
    }
]
```

untrash

Untrash files and directories

```
rclone backend untrash remote: [options] [<arguments>+]
```

This command untrashes all the files and directories in the directory passed in recursively.

Usage:

This takes an optional directory to trash which make this easier to use via the API.

```
rclone backend untrash drive:directory
rclone backend -i untrash drive:directory subdir
```

Use the -i flag to see what would be restored before restoring it.

Result:

```
{
    "Untrashed": 17,
    "Errors": 0
}
```

copyid

Copy files by ID

```
rclone backend copyid remote: [options] [<arguments>+]
```

This command copies files by ID

Usage:

```
rclone backend copyid drive: ID path
rclone backend copyid drive: ID1 path1 ID2 path2
```

It copies the drive file with ID given to the path (an rclone path which will be passed internally to rclone copyto). The ID and path pairs can be repeated.

The path should end with a / to indicate copy the file as named to this directory. If it doesn't end with a / then the last path component will be used as the file name.

If the destination is a drive backend then server-side copying will be attempted if possible.

Use the -i flag to see what would be copied before copying.

Limitations

Drive has quite a lot of rate limiting. This causes rclone to be limited to transferring about 2 files per second only. Individual files may be transferred much faster at 100s of MBytes/s but lots of small files can take a long time.

Server side copies are also subject to a separate rate limit. If you see User rate limit exceeded errors, wait at least 24 hours and retry. You can disable server-side copies with `--disable copy` to download and upload the files if you prefer.

Limitations of Google Docs

Google docs will appear as size -1 in `rclone ls` and as size 0 in anything which uses the VFS layer, e.g. `rclone mount`, `rclone serve`.

This is because rclone can't find out the size of the Google docs without downloading them.

Google docs will transfer correctly with `rclone sync`, `rclone copy` etc as rclone knows to ignore the size when doing the transfer.

However an unfortunate consequence of this is that you may not be able to download Google docs using `rclone mount`. If it doesn't work you will get a 0 sized file. If you try again the doc may gain its correct size and be downloadable. Whether it will work on not depends on the application accessing the mount and the OS you are running - experiment to find out if it does work for you!

Duplicated files

Sometimes, for no reason I've been able to track down, drive will duplicate a file that rclone uploads. Drive unlike all the other remotes can have duplicated files.

Duplicated files cause problems with the syncing and you will see messages in the log about duplicates.

Use `rclone dedupe` to fix duplicated files.

Note that this isn't just a problem with rclone, even Google Photos on Android duplicates files on drive sometimes.

Rclone appears to be re-copying files it shouldn't

The most likely cause of this is the duplicated file issue above - run `rclone dedupe` and check your logs for duplicate object or directory messages.

This can also be caused by a delay/caching on google drive's end when comparing directory listings. Specifically with team drives used in combination with --fast-list. Files that were uploaded recently may not appear on the directory list sent to rclone when using --fast-list.

Waiting a moderate period of time between attempts (estimated to be approximately 1 hour) and/or not using --fast-list both seem to be effective in preventing the problem.

Making your own client_id

When you use rclone with Google drive in its default configuration you are using rclone's client_id. This is shared between all the rclone users. There is a global rate limit on the number of queries per second that each client_id can do set by Google. rclone already has a high quota and I will continue to make sure it is high enough by contacting Google.

It is strongly recommended to use your own client ID as the default rclone ID is heavily used. If you have multiple services running, it is recommended to use an API key for each service. The default Google quota is 10 transactions per second so it is recommended to stay under that number as if you use more than that, it will cause rclone to rate limit and make things slower.

Here is how to create your own Google Drive client ID for rclone:

1. Log into the Google API Console with your Google account. It doesn't matter what Google account you use. (It need not be the same account as the Google Drive you want to access)

2. Select a project or create a new project.

3. Under "ENABLE APIS AND SERVICES" search for "Drive", and enable the "Google Drive API".

4. Click "Credentials" in the left-side panel (not "Create credentials", which opens the wizard), then "Create credentials"

5. If you already configured an "Oauth Consent Screen", then skip to the next step; if not, click on "CONFIGURE CONSENT SCREEN" button (near the top right corner of the right panel), then select "External" and click on "CREATE"; on the next screen, enter an "Application name" ("rclone" is OK) then click on "Save" (all other data is optional). Click again on "Credentials" on the left panel to go back to the "Credentials" screen.

(PS: if you are a GSuite user, you could also select "Internal" instead of "External" above, but this has not been tested/documented so far).

6. Click on the "+ CREATE CREDENTIALS" button at the top of the screen, then select "OAuth client ID".

7. Choose an application type of "Desktop app" if you using a Google account or "Other" if you using a GSuite account and click "Create". (the default name is fine)

8. It will show you a client ID and client secret. Use these values in rclone config to add a new remote or edit an existing remote.

Be aware that, due to the "enhanced security" recently introduced by Google, you are theoretically expected to "submit your app for verification" and then wait a few weeks(!) for their response; in practice, you can go right ahead and use the client ID and client secret with rclone, the only issue will be a very scary confirmation screen shown when you connect via your browser for rclone to be able to get its token-id (but as this only happens during the remote configuration, it's not such a big deal).

(Thanks to @balazer on github for these instructions.)

Sometimes, creation of an OAuth consent in Google API Console fails due to an error message “The request failed because changes to one of the field of the resource is not supported”. As a convenient workaround, the necessary Google Drive API key can be created on the Python Quickstart page. Just push the Enable the Drive API button to receive the Client ID and Secret. Note that it will automatically create a new project in the API Console.