Mega

Mega is a cloud storage and file hosting service known for its security feature where all files are encrypted locally before they are uploaded. This prevents anyone (including employees of Mega) from accessing the files without knowledge of the key used for encryption.

This is an rclone backend for Mega which supports the file transfer features of Mega using the same client side encryption.

Paths are specified as `remote:path`

Paths may be as deep as required, e.g. `remote:directory/subdirectory`.

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
s) Set configuration password
q) Quit config
n/s/q> n
name> remote
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Mega
   \ "mega"
[snip]
Storage> mega
User name
user> you@example.com
Password.
y) Yes type in my own password
g) Generate random password
n) No leave this optional password blank
y/g/n> y
Enter the password:
password:
Confirm the password:
password:
Remote config
--------------------
[remote]
type = mega
user = you@example.com
pass = *** ENCRYPTED ***
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

NOTE: The encryption keys need to have been already generated after a regular login via the browser, otherwise attempting to use the credentials in `rclone` will fail.

Once configured you can then use `rclone` like this,

List directories in top level of your Mega

```
rclone lsd remote:
```

List all the files in your Mega

```
rclone ls remote:
```

To copy a local directory to an Mega directory called backup

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

Modified time and hashes

Mega does not support modification times or hashes yet.

Restricted filename characters

Character	Value	Replacement
NUL	0x00	␀
/	0x2F	／

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

Duplicated files

Mega can have two files with exactly the same name and path (unlike a normal file system).

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

Use `rclone dedupe` to fix duplicated files.

Failure to log-in

Mega remotes seem to get blocked (reject logins) under "heavy use". We haven't worked out the exact blocking rules but it seems to be related to fast paced, successive rclone commands.

For example, executing this command 90 times in a row `rclone link remote:file` will cause the remote to become "blocked". This is not an abnormal situation, for example if you wish to get the public links of a directory with hundred of files... After more or less a week, the remote will remote accept rclone logins normally again.

You can mitigate this issue by mounting the remote it with `rclone mount`. This will log-in when mounting and a log-out when unmounting only. You can also run `rclone rcd` and then use `rclone rc` to run the commands over the API to avoid logging in each time.

Rclone does not currently close mega sessions (you can see them in the web interface), however closing the sessions does not solve the issue.

If you space rclone commands by 3 seconds it will avoid blocking the remote. We haven't identified the exact blocking rules, so perhaps one could execute the command 80 times without waiting and avoid blocking by waiting 3 seconds, then continuing...

Note that this has been observed by trial and error and might not be set in stone.

Other tools seem not to produce this blocking effect, as they use a different working approach (state-based, using sessionIDs instead of log-in) which isn't compatible with the current stateless rclone approach.

Note that once blocked, the use of other tools (such as megacmd) is not a sure workaround: following megacmd login times have been observed in succession for blocked remote: 7 minutes, 20 min, 30min, 30 min, 30min. Web access looks unaffected though.

Investigation is continuing in relation to workarounds based on timeouts, pacers, retrials and tpslimits - if you discover something relevant, please post on the forum.

So, if rclone was working nicely and suddenly you are unable to log-in and you are sure the user and the password are correct, likely you have got the remote blocked for a while.

Standard Options

Here are the standard options specific to mega (Mega).

--mega-user

User name

- Config: user
- Env Var: RCLONE_MEGA_USER
- Type: string
- Default: ""

--mega-pass

Password.

NB Input to this must be obscured - see rclone obscure.

- Config: pass
- Env Var: RCLONE_MEGA_PASS
- Type: string
- Default: ""

Advanced Options

Here are the advanced options specific to mega (Mega).

--mega-debug

Output more debug from Mega.

If this flag is set (along with -vv) it will print further debugging information from the mega backend.

- Config: debug
- Env Var: RCLONE_MEGA_DEBUG
- Type: bool
- Default: false

--mega-hard-delete

Delete files permanently rather than putting them into the trash.

Normally the mega backend will put all deletions into the trash rather than permanently deleting them. If you specify this then rclone will permanently delete objects instead.

- Config: hard_delete
- Env Var: RCLONE_MEGA_HARD_DELETE
- Type: bool
- Default: false

--mega-encoding

This sets the encoding for the backend.

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

- Config: encoding
- Env Var: RCLONE_MEGA_ENCODING
- Type: MultiEncoder
- Default: Slash,InvalidUtf8,Dot

Limitations

This backend uses the go-mega go library which is an opensource go library implementing the Mega API. There doesn't appear to be any documentation for the mega protocol beyond the mega C++ SDK source code so there are likely quite a few errors still remaining in this library.

Mega allows duplicate files which may confuse rclone.