Hubic

Paths are specified as `remote:path`

Paths are specified as `remote:container` (or `remote:` for the `lsd` command.) You may put subdirectories in too, e.g. `remote:container/path/to/dir`.

The initial setup for Hubic involves getting a token from Hubic 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:

```
n) New remote
s) Set configuration password
n/s> n
name> remote
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Hubic
   \ "hubic"
[snip]
Storage> hubic
Hubic Client Id - leave blank normally.
client_id>
Hubic Client Secret - leave blank normally.
client_secret>
Remote config
Use auto config?
 * Say Y if not sure
 * Say N if you are working on a remote or headless machine
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
--------------------
[remote]
client_id =
client_secret =
token = {"access_token":"XXXXXX"}
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

See the remote setup docs for how to set it up on a machine with no Internet browser available.

Note that rclone runs a webserver on your local machine to collect the token as returned from Hubic. 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.

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

List containers in the top level of your Hubic

```
rclone lsd remote:
```

List all the files in your Hubic

```
rclone ls remote:
```

To copy a local directory to an Hubic directory called backup

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

If you want the directory to be visible in the official *Hubic browser*, you need to copy your files to the `default` directory

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

--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.

Modified time

The modified time is stored as metadata on the object as `X-Object-Meta-Mtime` as floating point since the epoch accurate to 1 ns.

This is a de facto standard (used in the official python-swiftclient amongst others) for storing the modification time for an object.

Note that Hubic wraps the Swift backend, so most of the properties of are the same.

Standard Options

Here are the standard options specific to hubic (Hubic).

--hubic-client-id

OAuth Client Id Leave blank normally.

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

--hubic-client-secret

OAuth Client Secret Leave blank normally.

- Config: client_secret

- Env Var: RCLONE_HUBIC_CLIENT_SECRET
- Type: string
- Default: ""

Advanced Options

Here are the advanced options specific to hubic (Hubic).

--hubic-token

OAuth Access Token as a JSON blob.

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

--hubic-auth-url

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

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

--hubic-token-url

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

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

--hubic-chunk-size

Above this size files will be chunked into a _segments container.

Above this size files will be chunked into a _segments container. The default for this is 5GB which is its maximum value.

- Config: chunk_size
- Env Var: RCLONE_HUBIC_CHUNK_SIZE
- Type: SizeSuffix
- Default: 5G

--hubic-no-chunk

Don't chunk files during streaming upload.

When doing streaming uploads (e.g. using rcat or mount) setting this flag will cause the swift backend to not upload chunked files.

This will limit the maximum upload size to 5GB. However non chunked files are easier to deal with and have an MD5SUM.

Rclone will still chunk files bigger than chunk_size when doing normal copy operations.

- Config: no_chunk
- Env Var: RCLONE_HUBIC_NO_CHUNK
- Type: bool
- Default: false

--hubic-encoding

This sets the encoding for the backend.

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

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

Limitations

This uses the normal OpenStack Swift mechanism to refresh the Swift API credentials and ignores the expires field returned by the Hubic API.

The Swift API doesn't return a correct MD5SUM for segmented files (Dynamic or Static Large Objects) so rclone won't check or use the MD5SUM for these.