NAME

filescli - command line interface to static files services

SYNOPSIS

filescli [**-C** *config*] *subcommand args*

Subcommands:

upload [**-c** *collection-id*] [**-m** *mime-type*] **-t** *title* [**-p** *ispublishable*] [**-l** *license-type*] [**-L** *license-URL*] *local-path remote-path*

setid -c *collection-id remote-path*

getstate *remote-path*

publish -c *collection-id*

getpub *remote-path*

getweb *remote-path*

DESCRIPTION

filescli interacts with Static Files related services to:

- upload files
- assign a collection id to an uploaded file
- get the state of an uploaded file
- publish an uploaded file
- ownload a file from the Web or Publish environments

config is the name of a config file holding service URLs. This defaults to *config.yml* in the current directory. It is a YAML file that should look like this:

```
hosts:
  zebedee: http://localhost:8082
```

```
files: http://localhost:26900
download-publishing: http://localhost:23600
download-web: http://localhost:23601
identity: http://localhost:25600
upload: http://localhost:25100
```

upload

The **upload** subcommand uploads *local-path* to *remote-path. local-path* must be a regular file. *remote-path* includes the full directory on the remote side, but it must not be an absolute path. For example, you can say *dir/file.txt* but you can't say */dir/file.txt*. The directory isn't required, you can pass just a filename.

collection-id is an optional collection id you can assign to the uploaded file. If you do not assign a collection id here, you can use the **setid** subcommand later.

mime-type specifies the mime type to be recorded against the file. It is a string like 'application/json'. If you don't specify a mime type, one will be guessed based on the contents of the file. Guessing can be unreliable, so it is better always to give a mime type. (See *https://pkg.go.dev/net/http#DetectContentType*)

title is required.

ispublishable is a boolean 'true' or 'false', defaults to 'false.'

license-type is a string like 'MIT' or 'BSD'. It defaults to 'MIT'.

license-url is a link to a page describing the license. It defaults to *https://opensource.org/license/MIT*. Be careful that you keep *license-type* and *license-url* consistent. You can't change either one after the file has been uploaded.

setid

The **setid** subcommand assigns a collection id to *remote-path.* This will fail if a collection id has already been assigned.

getstate

The **getstate** subcommand prints the state of *remote-path* to stdout as a json document. The output is unformatted, but you can run it through jq(1) to get something like this:

```
{
  "path": "project/README.txt",
  "is_publishable": false,
  "collection_id": "abcde-12345",
  "title": "README file",
  "size_in_bytes": 53,
  "type": "text/plain",
  "licence": "MIT",
  "licence_url": "https://opensource.org/licenses/MIT",
  "state": "UPLOADED",
  "etag": "2cacd3d4af3ec8aac13c1a54214b8c32"
}
```

publish

The **publish** subcommand publishes all the files with the collection id *collection-id.*

getpub

The **getpub** subcommand retrieves *remote-path* from the Publish environment and prints it to stdout.

getweb

The **getweb** subcommand retrieves *remote-path* from the Web environment and prints it to stdout. If the file is very large, and the WriteTimeout on the Download Service is too low, and if the file has not been fully decrypted into S3 yet, you will get an unexpected EOF error. The solution is to set the WriteTimeout higher. The workaround is to wait a while until the file has been fully decrypted.

(To set the WriteTimeout higher in the download service, you must set *s.Server.WriteTimeout* in *service/external/external.go* in the *dp-download-service* repo to a value big enough to allow the whole transfer to finish. This can vary depending on the size of the file and the user's download bandwidth.)

ENVIRONMENT

Not all operations require credentials. For operations that do, the following environment variables are used:

IDENTITY_TOKEN
IDENTITY_EMAIL
IDENTITY_PASSWORD

If IDENTITY_TOKEN is set, it will be used directly. Otherwise, IDENTITY_EMAIL and IDENTITY_PASSWORD will be used to get an identity token from the *identity* host in the config file.

FLORENCE_TOKEN
FLORENCE_EMAIL
FLORENCE_PASSWORD

If FLORENCE_TOKEN is set, it will be used directly. Otherwise, FLORENCE_EMAIL and FLORENCE_PASSWORD will be used to get a florence token from the *zebedee* host in the config file.

FILES

config.yml

EXIT STATUS

filescli exits 0 on success, 2 on a usage error, and 1 on any other error.

EXAMPLES

Upload a file without setting a collection id:

```
$ filescli upload -m text/plain -t ’README file’ \
    README.txt project/README.txt
```

Set the file’s collection id:

```
$ filescli setid -c abcde-12345 project/README.txt
```

Get a file’s status:

```
$ filescli getstate project/README.txt | jq
{
  "path": "project/README.txt",
  "is_publishable": false,
  "collection_id": "abcde-12345",
  "title": "README file",
  "size_in_bytes": 53,
  "type": "text/plain",
  "licence": "MIT",
  "licence_url": "https://opensource.org/licenses/MIT",
  "state": "UPLOADED",
  "etag": "2cacd3d4af3ec8aac13c1a54214b8c32"
}
```

Download a file from the Publish environment:

```
$ filescli getpub project/README.txt
This is the content of README.txt as downloaded.....
```

Attempt to download an unpublished file from the Web enviroment:

```
$ filescli getweb project/README.txt
file not found on remote
```

Publish to the Web environment:

```
$ filescli publish -c abcde-12345
```

Now the download from the Web environment will work:

```
$ filescli getweb project/README.txt
This is the content of README.txt as downloaded.....
```

DIAGNOSTICS

You may see these messages if the IDENTITY_* and FLORENCE_* environment variables are not set:

```
could not get authentication token: identity server, email and password required
could not get login token: login server, email and password required
```

These messages are not necessarily a problem if the operation doesn't require credentials. In that case, the messages are just warnings.

If the operation requires credentials, these messages are saying there is not enough information provided to get the required tokens. You should set the appropriate environment variables.

SEE ALSO

https://github.com/ONSdigital/dp-static-files-compose