acd_cli Documentation

Release 0.3.1

yadayada

August 06, 2016

Contents

Version 0.3.1

Contents:

CHAPTER 1

Setting up acd_cli

Check which Python 3 version is installed on your system, e.g. by running

```
python3 -V
```

If it is Python 3.2.3, 3.3.0 or 3.3.1, you need to upgrade to a higher minor version.

You may now proceed to install using PIP, your Arch package manager or build Debian/RedHat packages.

1.1 Installation with PIP

If you are new to Python, worried about dependencies or about possibly messing up your system, create and activate virtualenv like so:

```
cd /parent/path/to/your/new/virtualenv
virtualenv acdcli
source acdcli/bin/activate
```

You are now safe to install and test acd_cli. When you are finished, the environment can be disabled by simply closing your shell or running `deactivate`.

Please check which pip command is appropriate for Python 3 packages in your environment. I will be using 'pip3' as superuser in the examples.

The recommended and most up-to-date way is to directly install the master branch from GitHub.

```
pip3 install --upgrade git+https://github.com/yadayada/acd_cli.git
```

The easiest way is to directly install from PyPI.

```
pip3 install --upgrade --pre acdcli
```

1.1.1 PIP Errors

A version incompatibility may arise with PIP when upgrading the requests package. PIP will throw the following error:

```
ImportError: cannot import name 'IncompleteRead'
```

Run these commands to fix it:

```
apt-get remove python3-pip
easy_install3 pip
```

This will remove the distribution's pip3 package and replace it with a version that is compatible with the newer requests package.

1.2 Installation on Arch/Debian/RedHat

1.2.1 Arch Linux

There are two packages for Arch Linux in the AUR, acd_cli-git, which is linked to the master branch of the GitHub repository, and acd_cli, which is linked to the PyPI release.

1.2.2 Building deb/rpm packages

You will need to have fpm installed to build packages.

There is a Makefile in the assets directory that includes commands to build Debian packages (`make deb`) or RedHat packages (`make rpm`). It will also build the required requests-toolbelt package. fpm may also be able to build packages for other distributions or operating systems.

1.3 Environment Variables

1.3.1 Cache Path and Settings Path

You will find the current path settings in the output of `acd_cli -v init`.

The cache path is where acd_cli stores OAuth data, the node cache, logs etc. You may override the cache path by setting the `ACD_CLI_CACHE_PATH` environment variable.

The settings path is where various configuration files are stored (refer to the configuration section). The default path may be overriden by setting the `ACD_CLI_SETTINGS_PATH` environment variable.

1.3.2 Proxy support

Requests supports HTTP(S) proxies via environment variables. Since all connections to Amazon Drive are using HTTPS, you need to set the variable `HTTPS_PROXY`. The following example shows how to do that in a bash-compatible environment.

```
export HTTPS_PROXY="https://user:pass@1.2.3.4:8080/"
```

You can also use HTTP proxies supporting CONNECT method:

```
export HTTPS_PROXY="http://1.2.3.4:8888/"
```

Another way to permanently set the proxy is via configuration file.

1.3.3 Locale

If you need non-ASCII file/directory names, please check that your system's locale is set correctly.

1.4 Dependencies

1.4.1 FUSE

For the mounting feature, fuse >= 2.6 is needed according to fusepy. On a Debian-based distribution, the package should be named simply 'fuse'.

1.4.2 Python Packages

Under normal circumstances, it should not be necessary to install the dependencies manually.

- appdirs
- colorama
- dateutils
- requests >= 2.1.0
- requests-toolbelt
- sqlalchemy

If you want to the dependencies using your distribution's packaging system and are using a distro based on Debian 'jessie', the necessary packages are `python3-appdirs python3-colorama python3-dateutil python3-requests python3-sqlalchemy`.

1.5 Uninstalling

Please run `acd_cli delete-everything` first to delete your authentication and node data in the cache path. Then, use pip to uninstall

```
pip3 uninstall acdcli
```

Then, revoke the permission for `acd_cli_oa` to access your drive in your Amazon profile, more precisely at https://www.amazon.com/ap/adam.

CHAPTER 2

Authorization

Before you can use the program, you will have to complete the OAuth procedure with Amazon. It is necessary to have a Web browser installed in order to do so. There is a fast and simple way and a secure way.

2.1 Simple (Appspot)

You will not have to prepare anything to initiate this authorization method, just run, for example, `acd_cli init`.

A browser (tab) will open and you will be asked to log into your Amazon account or grant access for 'acd_cli_oa'. Signing in or clicking on 'Continue' will download a JSON file named `oauth_data`, which must be placed in the cache directory displayed on screen (e.g. `/home/<USER>/.cache/acd_cli`).

You may view the source code of the Appspot app that is used to handle the server part of the OAuth procedure at https://tensile-runway-92512.appspot.com/src.

2.2 Advanced Users (Security Profile)

You must create a security profile and have it whitelisted. Have a look at Amazon's ACD getting started guide. Select all permissions for your security profile and add a redirect URL to `http://localhost`.

Put your own security profile data in a file called `client_data` in the cache directory and have it adhere to the following form.

```
{
    "CLIENT_ID": "amzn1.application-oa2-client.0123456789abcdef0123456789abcdef",
    "CLIENT_SECRET": "0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef"
}
```

You may now run `acd_cli -v init`. The authentication procedure is similar to the one above. A browser (tab) will be opened and you will be asked to log in. Unless you have a local webserver running on port 80, you will be redirected to your browser's error page. Just copy the URL (e.g. `http://localhost/?code=AbCdEfGhIjKlMnOpQrSt&scope=clouddrive%3Aread_all+clouddrive%3Awrite`) into the console.

2.3 Changing Authorization Methods

If you want to change between authorization methods, go to your cache path (it is stated in the output of `acd_cli -v init`) and delete the file `oauth_data` and, if it exists, `client_data`.

CHAPTER 3

Usage

acd_cli may be invoked as `acd_cli` or `acdcli`.

Most actions need the node cache to be initialized and up-to-date, so please run a sync. A sync will fetch the changes since the last sync or the full node list if the cache is empty.

The following actions are built in

```
sync (s)            refresh node list cache; necessary for many actions
clear-cache (cc)    clear node cache [offline operation]

tree (t)            print directory tree [offline operation]
children (ls)       list a folder's children [offline operation]

find (f)            find nodes by name [offline operation] [case insensitive]
find-md5 (fm)       find files by MD5 hash [offline operation]
find-regex (fr)     find nodes by regular expression [offline operation] [case insensitive]

upload (ul)         file and directory upload to a remote destination
overwrite (ov)      overwrite file A [remote] with content of file B [local]
stream (st)         upload the standard input stream to a file
download (dl)       download a remote folder or file; will skip existing local files
cat                 output a file to the standard output stream

create (c, mkdir)   create folder using an absolute path

list-trash (lt)     list trashed nodes [offline operation]
trash (rm)          move node to trash
restore (re)        restore node from trash

move (mv)           move node A into folder B
rename (rn)         rename a node

resolve (rs)        resolve a path to a node ID [offline operation]

usage (u)           show drive usage data
quota (q)           show drive quota [raw JSON]
metadata (m)        print a node's metadata [raw JSON]

mount               mount the drive at a local directory
umount              unmount drive(s)
```

Please run `acd_cli --help` to get a current list of the available actions. A list of further arguments of an action and their order can be printed by calling `acd_cli [action] --help`.

Most node arguments may be specified as a 22 character ID or a UNIX-style path. Trashed nodes' paths might not be able to be resolved correctly; use their ID instead.

There are more detailed instructions for file transfer actions, find actions and FUSE documentation.

Logs will automatically be saved into the cache directory.

3.1 Global Flags/Parameters

`--verbose` (`-v`) and `--debug` (`-d`) will print additional messages to standard error.

`--no-log` (`-nl`) will disable the automatic logging feature that saves log files to the cache directory.

`--color` will set the coloring mode according to the specified argument (`auto`, `never` or `always`). Coloring is turned off by default; it is used for file/folder listings.

`--check` (`-c`) sets the start-up database integrity check mode. The default is to perform a `full` check. Setting the check to `quick` or `none` may speed up the initialization for large databases.

`--utf` (`-u`) will force the output to be encoded in UTF-8, regardless of the system's settings.

3.2 Exit Status

When the script is done running, its exit status can be checked for flags. If no error occurs, the exit status will be 0. Possible flag values are:

flag	value
general error	1
argument error	2
failed file transfer	8
upload timeout	16
hash mismatch	32
error creating folder	64
file size mismatch	128
cache outdated	256
remote duplicate	512
duplicate inode	1024
name collision	2048
error deleting source file	4096

If multiple errors occur, their respective flag values will be compounded into the exit status value by a binary OR operation. Because exit status values may not be larger than 255, flags 256 and above cannot be returned via exit status. A warning message will be displayed at the end of execution if those errors occured.

CHAPTER 4

Configuration

Some module constants may be set in INI-style configuration files. If you want to override the defaults as described below, create a plain text file for the module using the section heading as the file name in the settings directory.

4.1 acd_client.ini

```
[endpoints]
filename = endpoint_data

;sets the validity of the endpoint URLs, 3 days by default [seconds]
validity_duration = 259200

[transfer]
;sets the read/write chunk size for the local file system [bytes]
fs_chunk_size = 131072

;sets maximal consecutive chunk size for downloads, 500MiB by default [bytes]
;this limit was introduced because, in the past, files >10GiB could not be downloaded in one piece
dl_chunk_size = 524288000

;sets the number of retries for failed chunk requests
chunk_retries = 1

;sets the connect and idle timeout [seconds]
;the idle timeout will be used in both timeout scenarios for some old requests versions
;refer to the requests docs http://docs.python-requests.org/en/master/user/advanced/
connection_timeout = 30
idle_timeout = 60

[proxies]
;none by default
```

A proxy may be set by adding a protocol to proxy mapping like `https = https://user:pass@1.1.1.1:1234` to the proxies section.

4.2 cache.ini

```
[sqlite]
filename = nodes.db
```

```
;sets the time to sleep if a table is locked [milliseconds]
busy_timeout = 30000

;https://www.sqlite.org/pragma.html#pragma_journal_mode
journal_mode = wal

[blacklist]

;files contained in folders in this list will be excluded from being saved
;into the cache (not currently implemented)
folders = []
```

4.3 fuse.ini

```
[read]
;maximal number of simultaneously opened chunks per file
open_chunk_limit = 10

;sets the connection/idle timeout when creating or reading a chunk [seconds]
timeout = 5

[write]
;number of buffered chunks in the write queue
;the size of the chunks may vary (e.g. 512B, 4KB, or 128KB)
buffer_size = 32

;sets the timeout for putting a chunk into the queue [seconds]
timeout = 30
```

CHAPTER 5

File transfer

acd_cli offers multi-file transfer actions - upload and download - and single-file transfer actions - overwrite, stream and cat.

Multi-file transfers can be done with concurrent connections by specifying the argument `-x NUM`. If remote folder hierarchies or local directory hierarchies need to be created, this will be done prior to the file transfers.

5.1 Actions

5.1.1 upload

The upload action will upload files or recursively upload directories. Existing files will not be changed, normally.

Syntax:

```
acdcli upload /local/path [/local/next_path [...]] /remote/path
```

If the `--overwrite` (`-o`) argument is specified, a remote file will be updated if a) the local file's modification time is higher or b) the local file's creation time is higher and the file size is different. The `--force` (`-f`) argument can be used to force overwrite.

Hint: When uploading large files (>10GiB), a warning about a timeout may be displayed. You then need to wait a few minutes, sync and manually check if the file was uploaded correctly.

5.1.2 overwrite

The upload action overwrites the content of a remote file with a local file.

Syntax:

```
acdcli overwrite /local/path /remote/path
```

5.1.3 download

The download action can download a single file or recursively download a directory. If a file already exists locally, it will not be overwritten.

Syntax:

```
acdcli download /remote/path [/local/path]
```

If the local path is omitted, the destination path will be the current working directory.

5.1.4 stream

This action will upload the standard input stream to a file.

Syntax:

```
some_process | acdcli stream file_name /remote/path
```

If the `--overwrite` (`-o`) argument is specified, the remote file will be overwritten if it exists.

5.1.5 cat

This action outputs the content of a file to standard output.

5.2 Hints

Abort/Resume

Incomplete file downloads will be resumed automatically. Aborted file uploads are not resumable at the moment.

Folder or directory hierarchies that were created for a transfer do not need to be recreated when resuming a transfer.

Retry

Failed upload, download and overwrite actions allow retries on error by specifying the `--max-retries|-r` argument, e.g. `acd_cli <ACTION> -r MAX_RETRIES`.

Exclusion

Files may be excluded from upload or download by regex on their name or by file ending. Additionally, paths can be excluded from upload. Regexes and file endings are case-insensitive.

It is possible to specify multiple exclusion arguments of the same kind.

Remove Source Files

The `--remove-source-files|-rsf` flag is used, local files will be deleted from the filesystem

1. if the upload succeeds
2. if deduplication is enabled and at least one duplicate is found
3. if a file of the same name is present in the remote upload path but the file is not to bei overwritten (deletion then only occurs if the file sizes match)

Deduplication

Server-side deduplication prevents completely uploaded files from being saved as a node if another file with the same MD5 checksum already exists. acd_cli can prevent uploading duplicates by checking local files' sizes and MD5s. Empty files are never regarded duplicates.

CHAPTER 6

Finding nodes

The find actions will search for normal (active) and trashed nodes and list them.

6.1 find

The find action will perform a case-insensitive search for files and folders that include the name or name segment given as argument, so e.g. `acdcli find foo` will find "foo" , "Foobar", etc.

6.2 find-md5

find-md5 will search for files that match the MD5 hash given. The location of a local file may be determined like so:

```
acdcli find-md5 `md5sum local/file | cut -d" " -f1`
```

6.3 find-regex

find-regex searches for the specified regex in nodes' names.

CHAPTER 7

FUSE module

7.1 Status

The FUSE module will never provide anything as good and reliable as a local filesystem. See the bug tracker for issues that may occur.

acd_cli's FUSE module has the following filesystem features implemented:

Feature	Working
Basic operations	
List directory	
Read	
Write	[1]
Rename	
Move	
Trashing	
OS-level trashing	[2]
View trash	
Misc	
Automatic sync	
ctime/mtime update	
Custom permissions	
Hard links	partially [3]
Symbolic links	[4]

7.2 Usage

The command to mount the (root of the) Amazon Drive to the empty directory `path/to/mountpoint` is

```
acd_cli -nl mount path/to/mountpoint
```

A non-root folder may be mounted similarly, by

```
acd_cli -nl mount --modules="subdir,subdir=/folder" path/to/mountpoint
```

[1] partial writes are not possible (i.e. writes at random offsets)
[2] restoring might not work
[3] manually created hard links will be displayed, but it is discouraged to use them
[4] soft links are not part of the ACD API

Unmounting is usually achieved by the following command

```
fusermount -u path/to/mountpoint
```

If the mount is busy, Linux users can use the `--lazy` (`-z`) flag. There exists a convenience action `acd_cli umount` that unmounts all ACDFuse mounts on Linux and Mac OS.

Note: Changes made to your Amazon Drive not using acd_cli will no longer be synchronized automatically. See the `--interval` option below to re-enable automatic synchronization.

Warning: Using acd_cli's CLI commands (e.g. upload or sync) while having the drive mounted may lead to errors or corruption of the node cache.

7.2.1 Mount Options

For further information on the most of the options below, see your *`mount.fuse(8)`* man page.

To convert the node's standard character set (UTF-8) to the system locale, the modules argument may be used, e.g. `--modules="iconv,to_code=CHARSET"`.

--allow-other, -ao allow all users to access the mountpoint (may need extra configuration)

--allow-root, -ar allow the root user to access the mountpoint (may need extra configuration)

--foreground, -fg do not detach process until filesystem is destroyed (blocks)

--gid GID override the group ID (defaults to the user's gid)

--interval INT, -i INT set the node cache sync (refresh) interval to INT seconds

--nlinks, -n calculate the number of links for folders (slower)

--nonempty, -ne allow mounting to a non-empty mount point

--read-only, -ro disallow write operations (does not affect cache refresh)

--single-threaded, -st disallow multi-threaded FUSE operations

--uid UID override the user ID (defaults to the user's uid)

--umask UMASK override the standard permission bits

7.2.2 Automatic Remount

It is advisable to wait until your network connection is up before you try to run the mount command.

Linux users may use the systemd service file from the assets directory to have the drive automatically remounted on login. Alternative ways are to add a crontab entry using the `@reboot` keyword or to add an fstab entry like so:

```
acdmount    /mount/point    fuse    _netdev    0    0
```

For this to work, an executable shell script /usr/bin/acdmount must be created

```
#!/bin/bash

acd_cli mount -nl $1
```

7.2.3 Library Path

If you want or need to override the standard libfuse path, you may set the environment variable *LIBFUSE_PATH* to the full path of libfuse, e.g.

```
export LIBFUSE_PATH="/lib/x86_64-linux-gnu/libfuse.so.2"
```

This is particularly helpful if the libfuse library is properly installed, but not found.

7.2.4 Deleting Nodes

"Deleting" directories or files from the file system will in reality trash them in Amazon Drive. Calling `rmdir` on a directory will always move it into the trash, even if it is not empty.

7.2.5 Logging

For debugging purposes, the recommended command to run is

```
acd_cli -d -nl mount -i0 -fg path/to/mountpoint
```

That command will disable the automatic refresh (i.e. sync) of the node cache (*-i0*) and disable detaching from the console.

CHAPTER 8

Contributing guidelines

8.1 Using the Issue Tracker

The issue tracker is not a forum! This does not mean there is no need for good etiquette, but that you should not post unnecessary information. Each reply will cause a notification to be sent to all of the issue's participants and some of them might consider it spam.

For minor corrections or additions, try to update your posts rather than writing a new reply. Use strike-through markdown for corrections and put updates at the bottom of your original post.

Please use the reaction button to "vote" on issues rather than commenting "+1" or similar.

8.1.1 Adding Issues

If you have a question, please read the documentation and search the issue tracker. If you still have a question, please consider using the Gitter chat or sending an e-mail to acd_cli@mail.com instead of opening an issue.

If you absolutely must open an issue, check that you are using the latest master commit and there is no existing issue that fits your problem (including closed and unresolved issues). Try to reproduce the issue on another machine or ideally on another operating system, if possible.

Please provide as much possibly relevant information as you can. This should at least contain:

- your operating system and Python version, e.g. as determined by `python3 -c 'import platform as p; print("%s\n%s" % (p.python_version(), p.platform()))'`
- the command/s you used
- what happened
- what you think should have happened instead (and maybe give a reason)

You might find the `--verbose` and, to a lesser extent, `--debug` flags helpful.

Caution: Be sure not to include authorization tokens from the log output in your comments.

Use code block markup for console output, log messages, etc.

8.2 Code

There are no real programming guidelines as of yet. Please use function annotations for typing like specified in PEP 3107 and, to stay 3.2-compliant, stringified PEP 484 type hints where appropriate. The limit on line length is 100

characters.

It is a generally a good idea to explicitly announce that you are working on a feature or an issue.

Please squash your commits and add yourself to the contributors list before making a pull request.

Have a look at Github's general guide how to contribute. It is not necessary to create a feature branch, i.e. you may commit to the master branch.

If you do not know how to contribute, look for issues tagged with "help wanted" and read the TODO list of some of the open tasks.

8.3 Donations

You might also want to consider making a donation to further the development of acd_cli.

CHAPTER 9

Contributors

Thanks to

- chrisidefix for adding the find-md5 action and forcing me to create a proper package and use PyPI
- msh100 for adding proxy documentation and updating the oauth scope
- hansendc for revamping the usage report
- legnaleurc for adding the find-regex action
- Timdawson264 for fixing st_nlinks in the FUSE node stat
- Lorentz83 for creating a bash completion script
- kylemanna for adding a systemd service file
- calisro for adding uid, gid, umask overrides for fuse layer
- memoz for amending proxy documentation

Also thanks to

- fibersnet for pointing out a possible deadlock in ACDFuse.
- and everyone else who I forgot to mention

CHAPTER 10

Frequently Asked Questions

10.1 Why Did I Get an UnicodeEncodeError?

If you encounter Unicode problems, check that your locale is set correctly. Alternatively, you may use the `--utf` argument to force acd_cli to use UTF-8 output encoding regardless of your console's current encoding.

Windows users may import the provided reg file (assets/win_codepage.reg), tested with Windows 8.1, to set the command line interface encoding to cp65001.

10.2 What Is acd_cli's Installation Path?

On unixoid operating systems the acd_cli script may be located by running `which acd_cli` or, if that does not yield a result, by executing `pip3 show -f acdcli`.

10.3 Where Does acd_cli Store its Cache and Settings?

You can see which paths are used in the log output of `acd_cli -v init`.

10.4 My Sync Fails. What Should I Do?

If you are doing an incremental synchronization (i.e. you have synchronized before) and it fails, a full sync might work `acd_cli sync -f`.

If the sync times out, consider increasing the idle timeout (refer to the config documentation).

You may also want to try the deprecated (and undocumented) synchronization method `acd_cli old-sync` if you happen to have only up to a few thousand files and folders in total.

10.5 How Do I Pass a Node ID Starting with – (dash/minus/hyphen)?

Precede the node ID by two minuses and a space to have it be interpreted as parameter and not as an argument, e.g. `-- -AbCdEfGhIjKlMnOpQr012`.

10.6 Can I Share or Delete Files/Folders?

No. It is not possible to share or delete using the Amazon Drive API. Please do it manually using the Web interface.

10.7 What Do I Do When I get an *sqlite3.OperationalError: database is locked* error?

Please limit the number or running acd_cli processes to one. For example, do not have an active FUSE mount while simultaneously uploading via command line.

10.8 Why Does Python Crash When executing acd_cli on Mac OS?

There is an issue with the _scproxy module. Please precede your usual commands by `env no_proxy='*'` to prevent it from causing crashes.

10.9 How Do I Share Directories from ACDFuse with Samba?

By default, only the user that originally mounted the FUSE filesystem has access permissions. To lift this restriction, run the `mount` command with the `--allow-other` option. You may need to edit your system's setting before being able to use this mount option, e.g. in /etc/fuse.conf.

10.10 Do Transfer Speeds Vary Depending on Geolocation?

Amazon may be throttling users not located in the U.S. To quote the Terms of Use,

> The Service is offered in the United States. We may restrict access from other locations. There may be limits on the types of content you can store and share using the Service, such as file types we don't support, and on the number or type of devices you can use to access the Service. We may impose other restrictions on use of the Service.

CHAPTER 11

Ancient History

11.1 0.1.3

- plugin mechanism added
- OAuth now via Appspot; security profile no longer necessary
- back-off algorithm for API requests implemented

11.2 0.1.2

new:

- overwriting of files
- recursive upload/download
- hashing of downloaded files
- clear-cache action

fixes:

- remove-child accepted status code
- fix for upload of files with Unicode characters

other:

- changed database schema

CHAPTER 12

Development

Contents:

12.1 acdcli package

12.1.1 Subpackages

acdcli.api package

Submodules

acdcli.api.account module

ACD account information

class acdcli.api.account.**AccountMixin**
Bases: object

fs_sizes() → tuple

Returns tuple total and free space

get_account_info() → dict
Gets account status [ACTIVE, ...?] and terms of use version.

get_account_usage() → str

get_quota() → dict

acdcli.api.backoff_req module

class acdcli.api.backoff_req.**BackOffRequest**(*auth_callback: 'requests.auth.AuthBase'*, *timeout: 'Tuple[int, int]'*, *proxies: dict={}*)
Bases: object

Wrapper for requests that implements timed back-off algorithm https://developer.amazon.com/public/apis/experience/cloud-drive/content/best-practices Caution: this catches all connection errors and may stall for a long time. It is necessary to init this module before use.

__init__(*auth_callback: 'requests.auth.AuthBase'*, *timeout: 'Tuple[int, int]'*, *proxies: dict={}*)

Parameters

- **auth_callback** – callable object that attaches auth info to a request
- **timeout** – tuple of connection timeout and idle timeout (http://docs.python-requests.org/en/latest/user/advanced/#timeouts)
- **proxies** – dict of protocol to proxy, see http://docs.python-requests.org/en/master/user/advanced/#proxies

delete(*url, acc_codes=[200], **kwargs*) → requests.models.Response

get(*url, acc_codes=[200], **kwargs*) → requests.models.Response

paginated_get(*url: str, params: dict=None*) → 'List[dict]'
Gets node list in segments of 200.

patch(*url, acc_codes=[200], **kwargs*) → requests.models.Response

post(*url, acc_codes=[200], **kwargs*) → requests.models.Response

put(*url, acc_codes=[200], **kwargs*) → requests.models.Response

acdcli.api.client module

class acdcli.api.client.**ACDClient**(*cache_path=''*, *settings_path=''*)
Bases: *acdcli.api.account.AccountMixin*, *acdcli.api.content.ContentMixin*, *acdcli.api.metadata.MetadataMixin*, *acdcli.api.trash.TrashMixin*

Provides a client to the Amazon Cloud Drive RESTful interface.

__init__(*cache_path=''*, *settings_path=''*)
Initializes OAuth and endpoints.

content_url

metadata_url

acdcli.api.common module

exception acdcli.api.common.**RequestError**(*status_code: int*, *msg: str*)
Bases: Exception

Catch-all exception class for various connection and ACD server errors.

class CODE
Bases: object

CONN_EXCEPTION = 1000

FAILED_SUBREQUEST = 1002

INCOMPLETE_RESULT = 1003

INVALID_TOKEN = 1005

REFRESH_FAILED = 1004

RequestError.**__init__**(*status_code: int*, *msg: str*)

RequestError.**codes = <lookup 'status_codes'>**

acdcli.api.common.**catch_conn_exception**(*func*)
Request connection exception decorator :raises RequestError

acdcli.api.common.**is_valid_id**(*id: str*) → bool

acdcli.api.content module

class acdcli.api.content.**ContentMixin**
Bases: object

Implements content portion of the ACD API.

chunked_download(**args*, ***kwargs*)

clear_file(*node_id: str*) → dict
Clears a file's content by overwriting it with an empty BytesIO.

Parameters **node_id** – valid file node ID

create_file(*file_name: str*, *parent: str=None*) → dict

create_folder(*name: str*, *parent=None*) → dict

download_chunk(*node_id: str*, *offset: int*, *length: int*, ***kwargs*) → bytearray
Load a file chunk into memory.

Parameters **length** – the length of the download chunk

download_file(*node_id: str*, *basename: str*, *dirname: str=None*, ***kwargs*)
Deals with download preparation, download with *chunked_download()* and finish. Calls callbacks while fast forwarding through incomplete file (if existent). Will not check for existing file prior to download and overwrite existing file on finish.

Parameters

- **dirname** – a valid local directory name, or cwd if None
- **basename** – a valid file name
- **kwargs** –
  - length: the total length of the file
  - write_callbacks (list[function]): passed on to *chunked_download()*
  - resume (bool=True): whether to resume if partial file exists

download_thumbnail(*node_id: str*, *file_name: str*, *max_dim=128*)
Download a movie's or picture's thumbnail into a file. Officially supports the image formats JPEG, BMP, PNG, TIFF, some RAW formats and the video formats MP4, QuickTime, AVI, MTS, MPEG, ASF, WMV, FLV, OGG. See http://www.amazon.com/gp/help/customer/display.html?nodeId=201634590 Additionally supports MKV.

Parameters **max_dim** – maximum width or height of the resized image/video thumbnail

overwrite_file(*node_id: str*, *file_name: str*, *read_callbacks: list=None*, *deduplication=False*) → dict

overwrite_stream(*stream*, *node_id: str*, *read_callbacks: list=None*) → dict
Overwrite content of node with ID *node_id* with content of *stream*.

Parameters **stream** – readable object

response_chunk(*node_id: str*, *offset: int*, *length: int*, ***kwargs*) → requests.models.Response

upload_file(*file_name: str*, *parent: str=None*, *read_callbacks=None*, *deduplication=False*) → dict

upload_stream(*stream*, *file_name: str*, *parent: str=None*, *read_callbacks=None*, *deduplication=False*) → dict

Parameters

- **stream** – readable object
- **parent** – parent node id, defaults to root node if None

acdcli.api.content.**PARTIAL_SUFFIX = '.__incomplete'**
suffix (file ending) for incomplete files

acdcli.api.metadata module

Node metadata operations

acdcli.api.metadata.**ChangeSet**
alias of Changes

class acdcli.api.metadata.**MetadataMixin**
Bases: object

add_child(*parent_id: str*, *child_id: str*) → dict
Adds node with ID *child_id* to folder with ID *parent_id*.

Returns updated child node dict

add_property(*node_id: str*, *owner_id: str*, *key: str*, *value: str*) → dict
Adds or overwrites *key* property with *content*. Maximum number of keys per owner is 10.

Parameters **value** – string of length <= 500

Raises RequestError: 404, <UnknownOperationException/> if owner is empty RequestError: 400, {...} if maximum of allowed properties is reached

Returns dict {'key': '<KEY>', 'location': '<NODE_ADDRESS>/properties/<OWNER_ID/<KEY>',

'value': '<VALUE>'}

delete_properties(*node_id: str*, *owner_id: str*)
Deletes all of the owner's properties. Uses multiple requests.

delete_property(*node_id: str*, *owner_id: str*, *key: str*)
Deletes *key* property from node with ID *node_id*.

get_asset_list() → list

get_changes(*checkpoint=''*, *include_purged=False*, *silent=True*, *file=None*)
Writes changes into a (temporary) file. See https://developer.amazon.com/public/apis/experience/cloud-drive/content/changes.

get_file_list() → list

get_folder_list() → list

get_metadata(*node_id: str*, *assets=False*, *temp_link=True*) → dict
Gets a node's metadata.

Parameters

- **assets** – also include asset info (e.g. thumbnails) if the node is a file
- **temp_link** – include a temporary download link if the node is a file

get_node_list(***params*) → list

Parameters **params** – may include tempLink='True'

get_owner_id()
Provisional function for retrieving the security profile's name, a.k.a. owner id.

get_root_id() → str
Gets the ID of the root node

Returns the topmost folder id

get_trashed_files() → list

get_trashed_folders() → list

list_children(*node_id: str*) → list

list_properties(*node_id: str*, *owner_id: str*) → dict
This will always return an empty dict if the accessor is not the owner. :param _sphinx_paramlinks_acdcli.api.metadata.MetadataMixin.list_properties.owner_id: owner ID (return status 404 if empty)

move_node(*node_id: str*, *parent_id: str*) → dict

move_node_from(*node_id: str*, *old_parent_id: str*, *new_parent_id: str*) → dict
Moves node with given ID from old parent to new parent. Not tested with multi-parent nodes.

Returns changed node dict

remove_child(*parent_id: str*, *child_id: str*) → dict

Returns updated child node dict

rename_node(*node_id: str*, *new_name: str*) → dict

set_available(*node_id: str*) → dict
Sets node status from 'PENDING' to 'AVAILABLE'.

update_metadata(*node_id: str*, *properties: dict*) → dict
Update a node's properties like name, description, status, parents, ...

acdcli.api.oauth module

class acdcli.api.oauth.**AppspotOAuthHandler**(*path*)
Bases: *acdcli.api.oauth.OAuthHandler*

APPSPOT_URL = 'https://tensile-runway-92512.appspot.com/'

__init__(*path*)

check_oauth_file_exists()
Checks for existence of oauth token file and instructs user to visit the Appspot page if it was not found.

Raises FileNotFoundError if oauth file was not placed into cache directory

refresh_auth_token()

Raises RequestError

class acdcli.api.oauth.**LocalOAuthHandler**(*path*)
Bases: *acdcli.api.oauth.OAuthHandler*

A local OAuth handler that works with a whitelisted security profile. The profile must not be created prior to June 2015. Profiles created prior to this month are not able to use the new scope "clouddrive:read_all" that replaces "clouddrive:read". https://developer.amazon.com/public/apis/experience/cloud-drive/content/getting-started

AMAZON_OA_LOGIN_URL = 'https://amazon.com/ap/oa'

AMAZON_OA_TOKEN_URL = 'https://api.amazon.com/auth/o2/token'

CLIENT_DATA_FILE = 'client_data'

REDIRECT_URI = 'http://localhost'

__init__(*path*)

check_oauth_file_exists()

Raises Exception

load_client_data()

Raises IOError if client data file was not found

Raises KeyError if client data file has missing key(s)

refresh_auth_token()

Raises RequestError

class acdcli.api.oauth.**OAuthHandler**(*path*)
Bases: requests.auth.AuthBase

class KEYS
Bases: object

ACC_TOKEN = 'access_token'

EXP_IN = 'expires_in'

EXP_TIME = 'exp_time'

REDIRECT_URI = 'redirect_uri'

REFR_TOKEN = 'refresh_token'

OAuthHandler.**OAUTH_DATA_FILE = 'oauth_data'**

OAuthHandler.**__init__**(*path*)

OAuthHandler.**check_oauth_file_exists**()
Checks for OAuth file existence and one-time initialize if necessary. Throws on error.

OAuthHandler.**exp_time**

OAuthHandler.**get_access_token_info**() → dict

Returns

int exp: expiration time in sec, str aud: client id user_id, app_id, iat (exp time)

OAuthHandler.**get_auth_token**(*reload=True*) → str
Gets current access token, refreshes if necessary.

Parameters **reload** – whether the oauth token file should be reloaded (external update)

OAuthHandler.**load_oauth_data**()
Loads oauth data file, validate and add expiration time if necessary

OAuthHandler.**refresh_auth_token**()
Fetches a new access token using the refresh token.

OAuthHandler.**treat_auth_token**(*time_: float*)
Adds expiration time to member OAuth dict using specified begin time.

classmethod OAuthHandler.**validate**(*oauth: str*) → dict
Deserialize and validate an OAuth string

Raises RequestError

OAuthHandler.**write_oauth_data**()
Dumps (treated) OAuth dict to file as JSON.

acdcli.api.oauth.**create_handler**(*path: str*)

acdcli.api.trash module

Node trashing and restoration. https://developer.amazon.com/public/apis/experience/cloud-drive/content/trash

class acdcli.api.trash.**TrashMixin**
Bases: object

list_trash() → list
Retrieves top-level trash list

move_to_trash(*node_id: str*) → dict

purge(*node_id: str*) → dict

restore(*node_id: str*) → dict

Module contents

ACD API

Usage

```
from api import client
acd_client = client.ACDClient()
root = acd_client.get_root_id()
children = acd_client.list_children(root)
for child in children:
    print(child['name'])
# ...
```

Node JSON Format This is the usual node JSON format for a file:

```
{
       'contentProperties': {'contentType': 'text/plain',
                             'extension': 'txt',
                             'md5': 'd41d8cd98f00b204e9800998ecf8427e',
                             'size': 0,
                             'version': 1},
       'createdBy': '<security-profile-nm>-<user>',
       'createdDate': '2015-01-01T00:00:00.00Z',
       'description': '',
       'eTagResponse': 'AbCdEfGhI01',
       'id': 'AbCdEfGhIjKlMnOpQr0123',
       'isShared': False,
       'kind': 'FILE',
       'labels': [],
       'modifiedDate': '2015-01-01T00:00:00.000Z',
       'name': 'empty.txt',
       'parents': ['0123AbCdEfGhIjKlMnOpQr'],
       'restricted': False,
```

```
        'status': 'AVAILABLE',
        'version': 1
}
```

The `modifiedDate` and `version` keys get updated each time the content or metadata is updated. `contentProperties['version']` gets updated on overwrite.

A folder's JSON looks similar, but it lacks the `contentProperties` dictionary.

`isShared` is set to `False` even when a node is actually shared.

Caution: ACD allows hard links for folders!

acdcli.api.**new_dau**()

acdcli.cache package

Submodules

acdcli.cache.cursors module

Cursor context managers

class acdcli.cache.cursors.**cursor**(*conn*)
Bases: object

__init__(*conn*)

class acdcli.cache.cursors.**mod_cursor**(*conn*)
Bases: object

__init__(*conn*)

acdcli.cache.db module

exception acdcli.cache.db.**IntegrityError**(*msg*)
Bases: Exception

__init__(*msg*)

class acdcli.cache.db.**NodeCache**(*cache_path: str=''*, *settings_path=''*, *check=0*)
Bases: *acdcli.cache.schema.SchemaMixin*, *acdcli.cache.query.QueryMixin*, *acdcli.cache.sync.SyncMixin*, *acdcli.cache.format.FormatterMixin*

IntegrityCheckType = {'none': 2, 'quick': 1, 'full': 0}
types of SQLite integrity checks

__init__(*cache_path: str=''*, *settings_path=''*, *check=0*)

integrity_check(*type_: {'none': 2, 'quick': 1, 'full': 0}*)
Performs a self-integrity check on the database.

remove_db_file() → bool
Removes database file.

acdcli.cache.format module

Formatters for query Bundle iterables. Capable of ANSI-type coloring using colors defined in `LS_COLORS`.

class acdcli.cache.format.**FormatterMixin**
Bases: object

file_entry(*file*, *long=False*, *size_bytes=False*) → str

static id_format(*nodes*) → 'Generator[str]'

long_id_format(*nodes*) → 'Generator[str]'

ls_format(*folder_id*, *folder_path=None*, *recursive=False*, *trash_only=False*, *trashed_children=False*, *long=False*, *size_bytes=False*) → 'Generator[str]'

path_format(*nodes*)

size_nlink_str(*node*, *size_bytes=False*)
Creates a right-justified size/nlinks string.

tree_format(*node*, *path*, *trash=False*, *dir_only=False*, *depth=0*, *max_depth=None*) → 'Generator[str]'
A simple tree formatter that indicates parentship by indentation (i.e. does not display graphical branches like **tree**).

acdcli.cache.format.**color_file**(*name: str*) → str
Colorizes a file name according to its file ending.

acdcli.cache.format.**color_path**(*path: str*) → str
Colorizes a path string.

acdcli.cache.format.**color_status**(*status*)
Creates a colored one-character status abbreviation.

acdcli.cache.format.**date_str**(*time_: datetime.datetime*) → str
Creates colored date string similar to the one in ls -l.

acdcli.cache.format.**init**(*color=0*)
Disables pre-initialized coloring if never mode specified or stdout is a tty.

Parameters **color** – the color mode to use, defaults to auto

acdcli.cache.query module

class acdcli.cache.query.**Node**(*row*)
Bases: object

__init__(*row*)

created

is_available

is_file

is_folder

is_trashed

modified

simple_name

class acdcli.cache.query.**QueryMixin**
Bases: object

calculate_usage()

childrens_names(*folder_id*) → 'List[str]'

file_size_exists(*size*) → bool

find_by_md5(*md5*) → 'List[Node]'

find_by_name(*name: str*) → 'List[Node]'

find_by_regex(*regex*) → 'List[Node]'

first_path(*node_id: str*) → str

get_child(*folder_id*, *child_name*) → 'Union[Node|None]'

get_conflicting_node(*name: str*, *parent_id: str*)
Finds conflicting node in folder specified by *parent_id*, if one exists.

get_file_count() → int

get_folder_count() → int

get_node(*id*) → 'Union[Node|None]'

get_node_count() → int

get_root_node()

list_children(*folder_id*, *trash=False*) → 'Tuple[List[Node], List[Node]]'

list_trashed_children(*folder_id*) → 'Tuple[List[Node], List[Node]]'

num_children(*folder_id*) → int

num_parents(*node_id*) → int

resolve(*path: str*, *trash=False*) → 'Union[Node|None]'

acdcli.cache.query.**datetime_from_string**(*dt: str*) → datetime.datetime

acdcli.cache.schema module

class acdcli.cache.schema.**SchemaMixin**
Bases: object

create_tables()

drop_all()

init()

acdcli.cache.sync module

Syncs Amazon Node API objects with SQLite database.

class acdcli.cache.sync.**SyncMixin**
Bases: object

Sync mixin to the *NodeCache*

insert_files(*files: list*)

insert_folders(*folders: list*)
Inserts list of folders into cache. Sets 'update' column to current date.

Parameters **folders** – list of raw dict-type folders

insert_node(*node: dict*)
Inserts single file or folder into cache.

insert_nodes(*nodes: list*, *partial=True*)
Inserts mixed list of files and folders into cache.

insert_parentage(*nodes: list*, *partial=True*)

remove_purged(*purged: list*)
Removes purged nodes from database

Parameters **purged** – list of purged node IDs

acdcli.cache.sync.**gen_slice**(*list_*, *length=100*)

acdcli.cache.sync.**placeholders**(*args*)

Module contents

acdcli.plugins package

Submodules

acdcli.plugins.template module

This is a template that you can use for adding custom plugins.

class acdcli.plugins.template.**TestPlugin**
Bases: *acdcli.plugins.Plugin*

MIN_VERSION = '0.3.1'

classmethod action(*args: argparse.Namespace*) → int
This is where the magic happens. Return a zero for success, a non-zero int for failure.

classmethod attach(*subparsers: argparse.ArgumentParser*, *log: list*, ***kwargs*)
Attaches this plugin to the top-level argparse subparser group :param subparsers the action subparser group :param log a list to put initialization log messages in

registry = {<class 'acdcli.plugins.template.TestPlugin'>}

Module contents

class acdcli.plugins.**Plugin**
Bases: object

Plugin base class. May be subject to changes.

MAX_VERSION = None

MIN_VERSION = None

static action(*args: argparse.Namespace*)

classmethod attach(*subparsers: argparse.ArgumentParser*, *log: list*, ***kwargs*)

classmethod check_version(*version: str*) → bool

registry = {<class 'acdcli.plugins.template.TestPlugin'>}

class acdcli.plugins.**RegisterLeafClasses**(*name*, *bases*, *nmspc*)
Bases: type

__init__(*name*, *bases*, *nmspc*)

acdcli.utils package

Submodules

acdcli.utils.conf module

acdcli.utils.conf.**get_conf**(*path*, *filename*, *default_conf: configparser.ConfigParser*) → configparser.ConfigParser

acdcli.utils.hashing module

class acdcli.utils.hashing.**IncrementalHasher**
Bases: object

__init__()

get_result() → str

hasher

update(*chunk*)

acdcli.utils.hashing.**hash_file**(*file_name: str*) → str

acdcli.utils.hashing.**hash_file_obj**(*fo*) → str

acdcli.utils.progress module

class acdcli.utils.progress.**FileProgress**(*total_sz: int*, *current: int=0*)
Bases: object

__init__(*total_sz: int*, *current: int=0*)

current

done()

reset()

status

total

update(*chunk*)

class acdcli.utils.progress.**MultiProgress**
Bases: object

Container that accumulates multiple FileProgress objects

__init__()

add(*progress: acdcli.utils.progress.FileProgress*)

end()

print_progress()

acdcli.utils.progress.**file_size_str**(*num: int*, *suffix='B'*) → str

acdcli.utils.progress.**speed_str**(*num: int*, *suffix='B'*, *time_suffix='/s'*) → str

acdcli.utils.progress.**time_str**(*num: float*) → str

acdcli.utils.threading module

class acdcli.utils.threading.**QueuedLoader**(*workers=1*, *print_progress=True*, *max_retries=0*)
Bases: object

Multi-threaded loader intended for file transfer jobs.

MAX_NUM_WORKERS = 8

MAX_RETRIES = 4

REFRESH_PROGRESS_INT = 0.3

__init__(*workers=1*, *print_progress=True*, *max_retries=0*)

add_jobs(*jobs: list*)

Parameters **jobs** – list of partials that return a RetryRetVal and have a pg_handler kwarg

start() → int
Starts worker threads and, if applicable, progress printer thread. :returns: accumulated return value

acdcli.utils.time module

acdcli.utils.time.**datetime_to_timestamp**(*dt: datetime.datetime*) → float

Module contents

12.1.2 Submodules

12.1.3 acdcli.acd_fuse module

12.1.4 Module contents

12.2 acdcli

12.3 TODO

12.3.1 General / API

- switch to multiprocessing (?)
- metalink support (?)

12.3.2 API

- support of node labels
- support for assets (?)
- favorite support (feature not yet announced officially)
- rip out the Appspot authentication handler
- fix upload of 0-byte streams

12.3.3 CLI

- unify the find action
- check symlink behavior for different Python versions (#95)

12.3.4 FUSE

- invalidate chunks of StreamedResponseCache (implement a time-out)
- respect flags when opening files
- use a filesystem test suite

12.3.5 File Transfer

- more sophisticated progress handler that supports offsets
- copy local mtime on upload (#58)
- add path exclusion by argument for download

12.3.6 User experience

- shell completion for remote directories (#127)
- even nicer help formatting
- log coloring

12.3.7 Tests

- cache methods
- more functional tests
- fuse module

12.3.8 Documentation

- write how-to on packaging plugins (sample setup.py)

CHAPTER 13

Overview

acd_cli provides a command line interface to Amazon Drive and allows Unix users to mount their drive using FUSE for read and (sequential) write access. It is currently in beta stage.

CHAPTER 14

Node Cache Features

- local caching of node metadata in an SQLite database
- addressing of remote nodes via a pathname (e.g. `/Photos/kitten.jpg`)
- file search

CHAPTER 15

CLI Features

- tree or flat listing of files and folders
- simultaneous uploads/downloads, retry on error
- basic plugin support

15.1 File Operations

- upload/download of single files and directories
- streamed upload/download
- folder creation
- trashing/restoring
- moving/renaming nodes

CHAPTER 16

Documentation

The full documentation is available at https://acd-cli.readthedocs.io.

CHAPTER 17

Quick Start

Have a look at the *known issues*, then follow the setup guide and authorize. You may then use the program as described in the usage guide.

CHAPTER 18

CLI Usage Example

In this example, a two-level folder hierarchy is created in an empty drive. Then, a relative local path `local/spam` is uploaded recursively using two connections.

```
$ acd_cli sync
  Getting changes...
  Inserting nodes..

$ acd_cli ls /
  [PHwiEv53QOKoGFGqYNl8pw] [A] /

$ acd_cli mkdir /egg/
$ acd_cli mkdir /egg/bacon/

$ acd_cli upload -x 2 local/spam/ /egg/bacon/
  [##################################]   100.0% of  100MiB  12/12  654.4KB/s

$ acd_cli tree
  /
      egg/
          bacon/
              spam/
                  sausage
                  spam
  [...]
```

The standard node listing format includes the node ID, the first letter of its status and its full path. Possible statuses are “AVAILABLE” and “TRASH”.

CHAPTER 19

Known Issues

It is not possible to upload files using Python 3.2.3, 3.3.0 and 3.3.1 due to a bug in the http.client module.

19.1 API Restrictions

- the current upload file size limit is 50GiB
- uploads of large files >10 GiB may be successful, yet a timeout error is displayed (please check the upload by syncing manually)
- storage of node names is case-preserving, but not case-sensitive (this should not concern Apple users)
- it is not possible to share or delete files

CHAPTER 20

Contribute

Have a look at the contributing guidelines.

CHAPTER 21

Recent Changes

21.1 0.3.1

- general improvements for FUSE
- FUSE write support added
- added automatic logging
- sphinx documentation added

21.2 0.3.0

- FUSE read support added

21.3 0.2.2

- sync speed-up
- node listing format changed
- optional node listing coloring added (for Linux or via LS_COLORS)
- re-added possibility for local OAuth

21.4 0.2.1

- curl dependency removed
- added job queue, simultaneous transfers
- retry on error

21.5 0.2.0

- setuptools support

- workaround for download of files larger than 10 GiB
- automatic resuming of downloads

a

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

Q

R

S

T

U

V

W