

---

# 115wangpan Documentation

*Release 0.7.4*

**Shichao An**

June 23, 2015



<b>1</b>	<b>115 Wangpan</b>	<b>1</b>
1.1	Features . . . . .	1
1.2	Installation . . . . .	1
1.3	Usage . . . . .	2
1.4	CLI commands . . . . .	2
<b>2</b>	<b>Contents</b>	<b>3</b>
2.1	Tutorial . . . . .	3
2.2	CLI Commands . . . . .	8
2.3	API Reference . . . . .	12
<b>3</b>	<b>Indices and tables</b>	<b>23</b>



---

## 115 Wangpan

---

115 Wangpan (115 or 115) is an unofficial Python API and SDK for 115.com. Supported Python versions are 2.6, 2.7, 3.3, 3.4.

- Documentation: <http://115wangpan.readthedocs.org>
- GitHub: <https://github.com/shichao-an/115wangpan>
- PyPI: <https://pypi.python.org/pypi/115wangpan/>

### 1.1 Features

- Authentication
- Persistent session
- Tasks management: BitTorrent and links
- Files management: uploading, downloading, searching, and editing

### 1.2 Installation

`libcurl` is required. Install dependencies before installing the python package:

Ubuntu:

```
$ sudo apt-get install build-essential libcurl4-openssl-dev python-dev
```

Fedora:

```
$ sudo yum groupinstall "Development Tools"  
$ sudo yum install libcurl libcurl-devel python-devel
```

Then, you can install with pip:

```
$ pip install 115wangpan
```

Or, if you want to install the latest from GitHub:

```
$ pip install git+https://github.com/shichao-an/115wangpan
```

## 1.3 Usage

```
>>> import u115
>>> api = u115.API()
>>> api.login('username@example.com', 'password')
True
>>> tasks = api.get_tasks()
>>> task = tasks[0]
>>> print task.name
-Saki- episode of side-A
>>> print task.status_human
TRANSFERRED
>>> print task.size_human
1.6 GiB
>>> files = task.list()
>>> files
[<File: 8 .mkv>]
>>> f = files[0]
>>> f.url
u'http://cdnuni.115.com/some-very-long-url.mkv'
>>> f.directory
<Directory: -Saki- episode of side-A>
>>> f.directory.parent
<Directory: >
```

## 1.4 CLI commands

- 115 down: for downloading files
- 115 up: for creating tasks from torrents and links

## 2.1 Tutorial

### 2.1.1 Login and credential file

To login with your account, simply call `u115.API.login` with username and password as argument.

```
>>> from u115 import API
>>> api = API()
>>> api.login('username@example.com', 'password')
True
```

You can also put username and password in the file `~/.115` as credentials with the following format:

```
[default]
username = username@example.com
password = defaultpassword

[another]
username = another@example.com
password = anotherpassword
```

Then, you can login without providing username and password, only specifying the section name (defaults to `default`).

```
>>> api.login(section='another')
True
```

To check whether API has logged:

```
>>> api.has_logged_in
True
```

### 2.1.2 Cookies

You can enable cookies to make a persistent session by providing `persistent (True)` and `cookies_filename` arguments.

```
>>> from u115 import API
>>> api = API(persistent=True, cookies_filename='cookies.txt')
>>> api.has_logged_in
False
>>> api.login(section='default')
```

```
>>> api.has_logged_in
True
>>> api.save_cookies()
>>> exit()
# Invoke Python interpreter again
>>> from u115 import API
>>> api = API(persistent=True, cookies_filename='cookies.txt')
>>> api.has_logged_in # Already logged in using cookies
True
```

You can also save cookies or load cookies explicitly.

```
>>> from u115 import API
>>> api = API()
>>> api.login()
True
# Do something ...
>>> api.save_cookies()
>>> exit()
# Now invoke Python interpreter again
>>> from u115 import API
>>> api = API()
>>> api.load_cookies()
>>> api.has_logged_in
True
```

### 2.1.3 Getting tasks

You can retrieve tasks using `u115.API.get_tasks()`. This function takes an optional argument, `count`, which defaults to 30. You can specify a smaller or larger integer value:

```
>>> tasks = api.get_tasks(60)
>>> tasks
[<Task: TV (320K+BK)>, <Task: Sword Art Online II - 10.mkv>, <Task: -Saki- episode of side-A>]
```

To get count of total number of existing tasks and quota for this month:

```
>>> api.task_count
3
>>> api.task_quota
27
```

To get the information of a specific task:

```
>>> task = tasks[1]
>>> task.add_time
datetime.datetime(2014, 9, 6, 23, 48, 58)
>>> task.status_human
'TRANSFERRED'
```

Normally, if it is a BitTorrent task and has been transferred, it has an associated directory which contains “lixian” (offline) files:

```
>>> task.is_directory
True
>>> task.is_transferred
True
>>> task.directory
```



```
<Directory: Sword Art Online II - 10.mkv>
>>> task.list() # or task.directory.list()
[<File: Sword Art Online II - 10.mkv>]
```

## 2.1.4 Creating BitTorrent tasks

You can upload a torrent and create a task on the fly:

```
>>> api.add_task_bt('~/downloads/Black_Bullet_OP.torrent')
True
>>> api.get_tasks()
[<Task: Black Bullet OP>, <Task: TV (320K+BK)>, ...]
```

Or you can edit the torrent and select files before submitting:

```
>>> u = api.add_task_bt('~/downloads/Black_Bullet_OP.torrent', select=True)
>>> files = u.files
[<TorrentFile: [*] black bullet.mp4>, <TorrentFile: [*] black bullet [commentary].mp4>, ...]
>>> files[0].unselect()
>>> files
[<TorrentFile: [ ] black bullet.mp4>, <TorrentFile: [*] black bullet [commentary].mp4>, ...]
>>> u.submit()
```

## 2.1.5 Creating URL tasks

You can create URL (link) tasks. HTTP, HTTPS, FTP, Magnet, and eD2k links are supported.

```
>>> api.add_task_url('http://example.com/file.txt')
>>> api.add_task_url('magnet:?xt=urn:sha1:YNCKHT.2=urn:sha1:TXGCZQT')
```

## 2.1.6 System directories

The account has its offline space that stores files transferred from tasks or uploaded by users. The root directory has access to all available files.

```
>>> api.root_directory
<Directory: >
>>> api.root_directory.list()
[<Directory: >]
>>> api.receiver_directory
<Directory: >
```

After your tasks are transferred, the files go to the downloads directory (*u115.API.downloads\_directory*). Its parent directory is the receiver directory (*u115.API.receiver\_directory*)

```
>>> api.downloads_directory
<Directory: >
```

If you have created BitTorrent tasks by uploading torrents, you will have a torrents directory (*u115.API.torrents\_directory*), which holds uploaded torrent files.

```
>>> api.torrents_directory
<Directory: >
```

## 2.1.7 Directory and files

You can list contents of a directory using `list()` method of a *Directory*.

```
>>> entries = api.downloads_directory.list()
>>> d = entries[0]
>>> d_entries = d.list()
>>> d_entries
[<Directory: BK>, <Directory: MP3>, <File: Cover.jpg>
>>> dd_entries = d_entries[0].list()
>>> dd_entries.list()
[<File: IMG_0003.jpg>, <File: IMG_0004.jpg>, <File: IMG_0005.jpg>, <File: IMG_0006.jpg>]
```

To get number of entries in a directory:

```
>>> d.count
45
```

The default behavior is to list 30 entries. You can list specified number of entries, either with or without `count`. The following two are equivalent:

```
>>> d.list(100)
>>> d.list(count=100)
```

List all files by passing `count` of itself:

```
>>> d.list(d.count)
```

There are other arguments that can be passed to `list()`, e.g. `order` (order of entries, which can be *user\_ptime* (default), *file\_size* and *file\_name*), `asc` (whether in ascending order), `show_dir` (whether to show directories). For full list of arguments, see `u115.Directory.list()`.

```
# List 100 smallest files
>>> d.list(count=100, order='file_size', asc=True)
# Do not include directories
>>> d.list(show_dir=False)
```

A directory has a parent directory, which can be accessed via `parent` attribute:

```
>>> d.parent
<Directory: >
>>> d.parent.parent
<Directory: >
# root directory has no parent
>>> d.root_directory.parent
# None
```

Delete the file or directory:

```
>>> f = d.list()[0]
>>> f.delete()
True
>>> f.is_deleted
True
```

## 2.1.8 Search directories and files

You can search directories and files by passing the keyword to `u115.API.search()`:

```
>>> api.search('manga')
[<Directory: [WOLF][Mangaka-san][01-12+OVA01-06][GB][720P][END]>, <File: [WOLF][Mangaka-san][01][GB]
```

By default, the search result limits to 30 items. To get more result, pass `count` to the methods:

```
>>> api.search('.mp4', count=100)
```

## 2.1.9 Download files

For offline files, you can retrieve download links:

```
>>> f = dd_entries.list()[0]
>>> f
<File: video.mov>
>>> f.url
u'http://cdnuni.115.com/very-long-name.mov'
>>> f.download()
      1%   14.3 MiB   437.3 KiB/s   0:53:45 ETA
```

There are also extra options for `download()`. The default behavior is downloading the file to the current working directory, resuming download progress if the file already exists (with the same path), and automatically retrying upon closed transfer until the file's download is finished.

```
# Download to path ~/Downloads
>>> f.download(path='~/Downloads')
# Do not show progres
>>> f.download(show_progress=False)
# Override existing file without resuming downloads
>>> f.download(resume=False)
```

## 2.1.10 Upload files

You can upload files to any offline directory (defaults to `downloads_directory`):

```
>>> p = api.downloads_directory.parent
>>> p
<Directory: >
>>> u = api.upload('/path/to/photo.jpg', directory=p)
>>> u
<File: photo.jpg>
```

The file you upload can be a torrent file, which you can open later and create a task from it:

```
>>> u = api.upload('/path/to/movie.torrent')
>>> u.is_torrent
True
>>> t = u.open_torrent()
>>> t.submit()
```

## 2.1.11 Moving files and directories

You can move files and directories within your offline space:

```
>>> files = api.downloads_directory.list()
>>> files
[<File: photo.jpg>, <Directory: album>]
>>> photo, album = files[0], files[1]
>>> photo
<File: photo.jpg>
>>> album
<Directory: album>
>>> photo.move(album)
>>> photo.directory
<Directory: album>
>>> album.list()
[<File: photo.jpg>]
```

### 2.1.12 Creating directories

You can create directories within your offline space:

```
>>> api.downloads_directory.list()
[<File: photo.jpg>]
>>> api.downloads_directory.mkdir('New Dir')
>>> api.downloads_directory.list()
[<Directory: New Dir>, <File: photo.jpg>]
```

### 2.1.13 Renaming files and directories

You can rename files and directories:

```
>>> f1
<File: photo.jpg>
>>> f1.edit('good.jpg')
>>> f1
<File: good.jpg>
```

### 2.1.14 Account and storage info

You can get user info:

```
>>> api.user_id
u'123456789'
>>> api.username
u'yourname@exmaple.com'
>>> api.get_user_info()
{'u'state': True, u'data': ...}
```

You can have an overview of your storage information:

```
>>> api.get_storage_info(human=True)
{'u'total': '152.3 GiB', u'used': '3.6 GiB'}
```

## 2.2 CLI Commands

115 is the CLI command that comes with this package. There are two sub-commands:

- 115 down: for downloading files
- 115 up: for creating tasks from torrents and links

## 2.2.1 115

115 mainly handles authentication and getting account information. The default action is to attempt to authenticate the API through one of ways discussed later.

### Authentication

115 needs to be authenticated every time it is invoked. You can pass username to `-u` option, and you will be prompted for password without echoing. Note that if cookies is enabled (discussed in the next section), it will bypass any specified username (`-u yourname`) or section (`-d section`) and use the cookies instead.

```
$ 115 -u yourname
Password:
```

If you don't want to be prompted for password every time, *set the credential file and section*. If the section is other than *default*, you specify the section name with `-d`:

```
$ 115 -d section_name
```

### Using cookies

You can use cookies to enable persistent session. The cookies file is `~/.115cookies`. If this file exists, 115 down and 115 up will automatically use cookies. Otherwise, you have to save the cookies first using the `-c` option:

```
$ 115 -d section_name -c
Cookies is saved.
$ 115 down -t
API has logged in from cookies.
...
```

If using the default section (`default`), `-d` can be omitted:

```
$ 115 -c
Cookies is saved.
```

If you do not want to use cookies any more, simply remove the `~/.115cookies` file.

### Account information

`-i` option can be used to print account information:

```
$ 115 -i
Username: yourname@example.com
User ID: 1234567
Used storage: 1.0 GiB
Total storage: 30 TiB
Task count: 10
Task quota: 1000
```

### Debug mode

-D can enable debug by setting the effective log level to DEBUG:

```
$ 115 -D -i
DEBUG:API:_debug:
  TYPE: Response
  FUNC: has_logged_in
  STATE: False
CONTENT: {'state': False, 'message': ''}

API has logged in from cookies.
DEBUG:API:_debug:
  TYPE: Request
  FUNC: _req_get_storage_info
  URL: http://115.com
METHOD: GET
PARAMS: {'ac': 'get_storage_info', '_': '12345678', 'ct': 'ajax'}
...
```

### Print help

You can print a comprehensive help message using the -h option:

```
$ 115 -h
```

## 2.2.2 115 down

115 down is for listing tasks and downloading files and directories in the default downloads directory (*u115.API.downloads\_directory/*).

### Listing tasks

Use -t to list task. The result output will contain a list of numbered tasks, each with status, size and name. The tasks are reversely ordered by creation time.

```
$ 115 down -t
1 [TRANSFERRED] [3.2 GiB] <Task: [Airota&DHR&Mony][Jinrui wa Suitai Shimashita][1280x720]>
2 [TRANSFERRED] [3.7 GiB] <Task: [CASO][Kill_Me_Baby][01-13][GB_BIG5][1280x720][x264_AAC]>
3 [TRANSFERRED] [38.1 GiB] <Task: Shinryaku Ika Musume>
...
```

### Downloading files

The default behavior of 115 down is to list entries (defaults to 30 entries) in the default downloads directory.

```
$ 115 down
1 <Directory: [AirotaDHRMony][Jinrui wa Suitai Shimashita][1280x720]>
2 <Directory: [CASO][Kill_Me_Baby][01-13][GB_BIG5][1280x720][x264_AAC]>
3 <Directory: Shinryaku Ika Musume>
4 <Directory: [Kamigami] Psycho-Pass 01-22 [1280x720 x264 AAC Sub(Chi,Jap)]>
...
```

To list more entries than default, pass a number to the -n option:

```
$ 115 down -n 60
```

To further list contents in a numbered entry:

```
$ 115 down 1
<Directory: [AirotaDHRMony][Jinrui wa Suitai Shimashita][1280x720]> (13 out of 13)
  1 [291.2 MiB] <File: [Airota&DHR&Mony][Jinrui wa Suitai Shimashita][01][1280x720][x264_7]
  2 [277.5 MiB] <File: [Airota&DHR&Mony][Jinrui wa Suitai Shimashita][02][1280x720][x264_7]
  ...
```

To download all items of the specified entry, pass star (\*) as the second arguments after the entry number.

```
$ 115 down 1 \*
$ 115 down 1 '*'
```

To download only one, or a range of items of the specified entry, use a number or range of numbers, like in the following examples.

Download the first item:

```
$ 115 down 1 1
```

Download item 2 and 4:

```
$ 115 down 1 2,4
```

Download items 1 to 8:

```
$ 115 down 1 1-8
```

Download a combination of items:

```
$ 115 down 1 1,3-4,6,9-12
```

The default downloading behavior is keeping the directory structure. If you want to flatten the directory, pass the `-f` switch. This will download everything of this entry into the current working directory without creating any directories:

```
$ 115 down -f 1 `*`
```

If you want to print the files to be downloaded instead of really downloading them, use `-s` option to make a dry run.

```
$ 115 down -s 1 \*
```

### 2.2.3 115 up

You can create either BitTorrent or URL tasks using `115 up`.

To create a BitTorrent task, pass the torrent path to `-t`. If the task is successfully created, its name and status will be printed.

```
$ 115 up -t ~/torrents/Mangaka-san.torrent
Task is successfully created.
[WOLF][Mangaka-san][01-12+OVA01-06][GB][720P][END] BEING TRANSFERRED
```

To create a URL pass, pass the link to `-l`:

```
$ 115 up -l 'magnet:?xt=urn:btih...announce'
Task is successfully created.
[WOLF][Mangaka-san][01-12+OVA01-06][GB][720P][END] BEING TRANSFERRED
```

## 2.3 API Reference

### 2.3.1 Main interface

**class** `u115.API` (*persistent=False, cookies\_filename=None, cookies\_type='LWP CookieJar'*)

Request and response interface

#### Variables

- **passport** – *Passport* object associated with this interface
- **http** – *RequestHandler* object associated with this interface
- **num\_tasks\_per\_page** (*int*) – default number of tasks per page/request
- **web\_api\_url** (*str*) – files API url
- **aps\_natsort\_url** (*str*) – natural sort files API url
- **proapi\_url** (*str*) – pro API url for downloads

**\_\_init\_\_** (*persistent=False, cookies\_filename=None, cookies\_type='LWP CookieJar'*)

#### Parameters

- **auto\_logout** (*bool*) – whether to logout automatically when *API* object is destroyed  
Deprecated since version 0.6.0: Call *API.logout()* explicitly
- **persistent** (*bool*) – whether to use persistent session that stores cookies on disk
- **cookies\_filename** (*str*) – path to the cookies file, use default path (*~/115cookies*) if *None*
- **cookies\_type** (*str*) – a string representing *cookieilib.FileCookieJar* subclass, *LWP CookieJar* (default) or *MozillaCookieJar*

**add\_task\_bt** (*filename, select=False*)

Add a new BT task

#### Parameters

- **filename** (*str*) – path to torrent file to upload
- **select** (*bool*) – whether to select files in the torrent.
  - **True:** it returns the opened torrent (*Torrent*) and can then iterate files in *Torrent.files* and select/unselect them before calling *Torrent.submit()*
  - **False:** it will submit the torrent with default selected files

**add\_task\_url** (*target\_url*)

Add a new URL task

**Parameters** *target\_url* (*str*) – the URL of the file that to be downloaded

**aps\_natsort\_url** = 'http://aps.115.com/natsort/files.php'

**cookies**

Cookies of the current API session (cookies getter shortcut)

**download** (*obj, path=None, show\_progress=True, resume=True, auto\_retry=True, proapi=False*)

Download a file

#### Parameters



- **obj** – *File* object
- **path** (*str*) – local path
- **show\_progress** (*bool*) – whether to show download progress
- **resume** (*bool*) – whether to resume on unfinished downloads identified by filename
- **auto\_retry** (*bool*) – whether to retry automatically upon closed transfer until the file's download is finished
- **proapi** (*bool*) – whether to use pro API

**downloads\_directory**

Default directory for downloaded files

**edit** (*entry, name, mark=False*)

Edit an entry (file or directory)

**Parameters**

- **entry** – BaseFile object
- **name** (*str*) – new name for the entry
- **mark** (*bool*) – whether to bookmark the entry

**get\_storage\_info** (*human=False*)

Get storage info

**Parameters** **human** (*bool*) – whether return human-readable size

**Returns** total and used storage

**Return type** *dict*

**get\_tasks** (*count=30*)

Get *count* number of tasks

**Parameters** **count** (*int*) – number of tasks to get

**Returns** a list of *Task* objects

**get\_user\_info** ()

Get user info

**Returns** a dictionary of user information

**Return type** *dict*

**has\_logged\_in**

Check whether the API has logged in

**load\_cookies** (*ignore\_discard=True, ignore\_expires=True*)

Load cookies from the file `API.cookies_filename`

**login** (*username=None, password=None, section='default'*)

Created the passport with `username` and `password` and log in. If either `username` or `password` is `None` or omitted, the credentials file will be parsed.

**Parameters**

- **username** (*str*) – username to login (email, phone number or user ID)
- **password** (*str*) – password
- **section** (*str*) – section name in the credential file

**Raise** raises *AuthenticationError* if failed to login

**logout** ()

Log out

**mkdir** (*parent, name*)

Create a directory

**Parameters**

- **parent** – the parent directory
- **name** (*str*) – the name of the new directory

**Returns** the new directory

**Return type** *Directory*

**move** (*entries, directory*)

Move one or more entries (file or directory) to the destination directory

**Parameters**

- **entries** (*list*) – a list of source entries (*BaseFile* object)
- **directory** – destination directory

**Returns** whether the action is successful

**Raise** *APIError* if something bad happened

**num\_tasks\_per\_page** = 30

**proapi\_url** = 'http://proapi.115.com/app/chrome/down'

**receiver\_directory**

Parent directory of the downloads directory

**referer\_url** = 'http://115.com'

**root\_directory**

Root directory

**save\_cookies** (*ignore\_discard=True, ignore\_expires=True*)

Save cookies to the file *API.cookies\_filename*

**search** (*keyword, count=30*)

Search files or directories

**Parameters**

- **keyword** (*str*) – keyword
- **count** (*int*) – number of entries to be listed

**task\_count**

Number of tasks created

**task\_quota**

Task quota (monthly)

**torrents\_directory**

Default directory that stores uploaded torrents

**upload** (*filename, directory=None*)

Upload a file *filename* to *directory*

**Parameters**

- **filename** (*str*) – path to the file to upload
- **directory** – destination *Directory*, defaults to **:at-tribute:'.API.downloads\_directory'** if None

**Returns** the uploaded file

**Return type** *File*

**user\_id**

User id of the current API user

**username**

Username of the current API user

**web\_api\_url** = 'http://web.api.115.com/files'

## 2.3.2 Request and response

**class** `u115.RequestHandler`

Request handler that maintains session

**Variables** **session** – underlying `requests.Session` instance

**get** (*url*, *params=None*)

Initiate a GET request

**post** (*url*, *data*, *params=None*)

Initiate a POST request

**send** (*request*, *expect\_json=True*, *ignore\_content=False*)

Send a formatted API request

**Parameters**

- **request** (*Request*) – a formatted request object
- **expect\_json** (*bool*) – if True, raise *InvalidAPIAccess* if response is not in JSON format
- **ignore\_content** (*bool*) – whether to ignore setting content of the Response object

**class** `u115.Request` (*url*, *method='GET'*, *params=None*, *data=None*, *files=None*, *headers=None*)

Formatted API request class

**\_\_init\_\_** (*url*, *method='GET'*, *params=None*, *data=None*, *files=None*, *headers=None*)

Create a Request object

**Parameters**

- **url** (*str*) – URL
- **method** (*str*) – request method
- **params** (*dict*) – request parameters
- **data** (*dict*) – form data
- **files** (*dict*) – multipart form data
- **headers** (*dict*) – custom request headers

**class** `u115.Response` (*state*, *content*)

Formatted API response class

**Variables**

- **state** (*bool*) – whether API access is successful
- **content** (*dict*) – result content

### 2.3.3 Cookies

```
class u115.RequestsLWPCookieJar (filename=None, delayload=False, policy=None)
    requests.cookies.RequestsCookieJar compatible cookielib.LWPCookieJar
```

```
class u115.RequestsMozillaCookieJar (filename=None, delayload=False, policy=None)
    requests.cookies.RequestsCookieJar compatible cookielib.MozillaCookieJar
```

### 2.3.4 Authentication

```
class u115.Passport (username, password)
    Passport for user authentication
```

#### Variables

- **username** (*str*) – username
- **password** (*str*) – user password
- **form** (*dict*) – a dictionary of POST data to login
- **user\_id** (*int*) – user ID of the authenticated user
- **data** (*dict*) – data returned upon login

### 2.3.5 Task and file components

```
class u115.Task (api, add_time, file_id, info_hash, last_update, left_time, move, name, peers, percent_done,
    rate_download, size, status, cid, pid)
    BitTorrent or URL task
```

#### Variables

- **add\_time** (*datetime.datetime*) – added time
- **cid** (*str*) – associated directory id, if any. For a directory task ( e.g. BT task), this is its associated directory's cid. For a file task (e.g. HTTP url task), this is the cid of the downloads directory. This value may be None if the task is failed and has no corresponding directory
- **file\_id** (*str*) – equivalent to *cid* of *Directory*. This value may be None if the task is failed and has no corresponding directory
- **info\_hash** (*str*) – hashed value
- **last\_update** (*datetime.datetime*) – last updated time
- **left\_time** (*int*) – left time ()
- **move** (*int*) – moving state
  - 0: not transferred
  - 1: transferred
  - 2: partially transferred

- **name** (*str*) – name of this task
- **peers** (*int*) – number of peers
- **percent\_done** (*int*) – <=100, originally named *percentDone*
- **rate\_download** (*int*) – download rate (B/s), originally named *rateDownload*
- **size** (*int*) – size of task
- **size\_human** (*str*) – human-readable size
- **status** (*int*) – status code
  - -1: failed
  - 1: downloading
  - 2: downloaded
  - 4: searching resources

**count**

Number of entries in the associated directory

**delete** ()

Delete task (does not influence its corresponding directory)

**Returns** whether deletion is successful

**Raise** *TaskError* if the task is already deleted

**directory**

Associated directory, if any, with this task

**is\_bt**

Alias of *is\_directory*

**is\_deleted**

**Returns** whether this task is deleted

**Return type** *bool*

**is\_directory**

**Returns** whether this task is associated with a directory

**Return type** *bool*

**is\_transferred**

**Returns** whether this tasks has been transferred

**Return type** *bool*

**list** (*count=30, order='user\_ptime', asc=False, show\_dir=True, natsort=True*)

List files of the associated directory to this task.

**Parameters**

- **count** (*int*) – number of entries to be listed
- **order** (*str*) – originally named *o*
- **asc** (*bool*) – whether in ascending order
- **show\_dir** (*bool*) – whether to show directories

**parent**

Parent directory of the associated directory

**status\_human**

Human readable status

**Returns**

- *DOWNLOADING*: the task is downloading files
- *BEING TRANSFERRED*: the task is being transferred
- *TRANSFERRED*: the task has been transferred to downloads directory
- *SEARCHING RESOURCES*: the task is searching resources
- *FAILED*: the task is failed
- *DELETED*: the task is deleted
- *UNKNOWN STATUS*

**Return type** `str`

**class** `u115.Torrent` (*api, name, size, info\_hash, file\_count, files=None, \*args, \*\*kwargs*)

Opened torrent before becoming a task

**Variables**

- **api** – associated API object
- **name** (*str*) – task name, originally named *torrent\_name*
- **size** (*int*) – task size, originally named *torrent\_size*
- **info\_hash** (*str*) – hashed value
- **file\_count** (*int*) – number of files included
- **files** (*list*) – files included (list of *TorrentFile*), originally named *torrent\_filelist\_web*

**selected\_files**

List of selected *TorrentFile* objects of this torrent

**submit** ()

Submit this torrent and create a new task

**unselected\_files**

List of unselected *TorrentFile* objects of this torrent

**class** `u115.TorrentFile` (*torrent, path, size, selected, \*args, \*\*kwargs*)

File in the torrent file list

**Parameters**

- **torrent** (*Torrent*) – the torrent that holds this file
- **path** (*str*) – file path in the torrent
- **size** (*int*) – file size
- **selected** (*bool*) – whether this file is selected

**select** ()

Select this file

**unselect** ()  
Unselect this file

**class** `u115.File` (*api*, *fid*, *cid*, *name*, *size*, *file\_type*, *sha*, *date\_created*, *thumbnail*, *pickcode*, \**args*, \*\**kwargs*)  
File in a directory

#### Variables

- **fid** (*int*) – file id
- **cid** (*str*) – cid of the current directory
- **size** (*int*) – size in bytes
- **size\_human** (*str*) – human-readable size
- **file\_type** (*str*) – originally named *ico*
- **sha** (*str*) – SHA1 hash
- **date\_created** (*datetime.datetime*) – in “%Y-%m-%d %H:%M:%S” format, originally named *t*
- **thumbnail** (*str*) – thumbnail URL, originally named *u*
- **pickcode** (*str*) – originally named *pc*

**delete** ()  
Delete this file or directory

**Returns** whether deletion is successful

**Raise** *APIError* if this file or directory is already deleted

**directory**  
Directory that holds this file

**download** (*path=None*, *show\_progress=True*, *resume=True*, *auto\_retry=True*, *proapi=False*)  
Download this file

**edit** (*name*, *mark=False*)  
Edit this file or directory

#### Parameters

- **name** (*str*) – new name for this entry
- **mark** (*bool*) – whether to bookmark this entry

**get\_download\_url** (*proapi=False*)  
Get this file’s download URL

**Parameters** **proapi** (*bool*) – whether to use pro API

**is\_deleted**  
Whether this file or directory is deleted

**is\_torrent**  
Whether the file is a torrent

**move** (*directory*)  
Move this file or directory to the destination directory

**Parameters** **directory** – destination directory

**Returns** whether the action is successful

**Raise** *APIError* if something bad happened

**open\_torrent** ()

Open the torrent (if it is a torrent)

**Returns** opened torrent

**Return type** *Torrent*

**reload** ()

Reload file info and metadata

- **name**
- **sha**
- **pickcode**

**url**

Alias for *File.get\_download\_url* () with *proapi=False*

**class** `u115.Directory` (*api, cid, name, pid, count=-1, date\_created=None, pickcode=None, is\_root=False, \*args, \*\*kwargs*)

**Variables**

- **cid** (*str*) – cid of this directory
- **pid** (*str*) – represents the parent directory it belongs to
- **count** (*int*) – number of entries in this directory
- **date\_created** (*datetime.datetime*) – integer, originally named *t*
- **pickcode** (*str*) – string, originally named *pc*

**count**

Number of entries in this directory

**delete** ()

Delete this file or directory

**Returns** whether deletion is successful

**Raise** *APIError* if this file or directory is already deleted

**edit** (*name, mark=False*)

Edit this file or directory

**Parameters**

- **name** (*str*) – new name for this entry
- **mark** (*bool*) – whether to bookmark this entry

**is\_deleted**

Whether this file or directory is deleted

**is\_root**

Whether this directory is the root directory

**list** (*count=30, order='user\_ptime', asc=False, show\_dir=True, natsort=True*)

List directory contents

**Parameters**

- **count** (*int*) – number of entries to be listed



- **order** (*str*) – order of entries, originally named *o*. This value may be one of *user\_ptime* (default), *file\_size* and *file\_name*
- **asc** (*bool*) – whether in ascending order
- **show\_dir** (*bool*) – whether to show directories
- **natsort** (*bool*) – whether to use natural sort

Return a list of *File* or *Directory* objects

**max\_entries\_per\_load** = 24

**mkdir** (*name*)

Create a new directory in this directory

**move** (*directory*)

Move this file or directory to the destination directory

**Parameters** **directory** – destination directory

**Returns** whether the action is successful

**Raise** *APIError* if something bad happened

**parent**

Parent directory that holds this directory

**reload** ()

Reload directory info and metadata

•*name*

•*pid*

•*count*

### 2.3.6 Exceptions

**class** `u115.APIError` (*\*args*, *\*\*kwargs*)

General error related to API

**class** `u115.TaskError` (*\*args*, *\*\*kwargs*)

Task has unstable status or no directory operation

**class** `u115.AuthenticationError` (*\*args*, *\*\*kwargs*)

Authentication error

**class** `u115.InvalidAPIAccess` (*\*args*, *\*\*kwargs*)

Invalid and forbidden API access

**class** `u115.RequestFailure` (*\*args*, *\*\*kwargs*)

Request failure

**class** `u115.JobError` (*\*args*, *\*\*kwargs*)

Job running error (request multiple similar jobs simultaneously)



---

## Indices and tables

---

- `genindex`
- `search`



## Symbols

`__init__()` (u115.API method), 12  
`__init__()` (u115.Request method), 15

### A

`add_task_bt()` (u115.API method), 12  
`add_task_url()` (u115.API method), 12  
API (class in u115), 12  
APIError (class in u115), 21  
`aps_natsort_url` (u115.API attribute), 12  
AuthenticationError (class in u115), 21

### C

`cookies` (u115.API attribute), 12  
`count` (u115.Directory attribute), 20  
`count` (u115.Task attribute), 17

### D

`delete()` (u115.Directory method), 20  
`delete()` (u115.File method), 19  
`delete()` (u115.Task method), 17  
Directory (class in u115), 20  
`directory` (u115.File attribute), 19  
`directory` (u115.Task attribute), 17  
`download()` (u115.API method), 12  
`download()` (u115.File method), 19  
`downloads_directory` (u115.API attribute), 13

### E

`edit()` (u115.API method), 13  
`edit()` (u115.Directory method), 20  
`edit()` (u115.File method), 19

### F

File (class in u115), 19

### G

`get()` (u115.RequestHandler method), 15  
`get_download_url()` (u115.File method), 19  
`get_storage_info()` (u115.API method), 13

`get_tasks()` (u115.API method), 13  
`get_user_info()` (u115.API method), 13

### H

`has_logged_in` (u115.API attribute), 13

### I

InvalidAPIAccess (class in u115), 21  
`is_bt` (u115.Task attribute), 17  
`is_deleted` (u115.Directory attribute), 20  
`is_deleted` (u115.File attribute), 19  
`is_deleted` (u115.Task attribute), 17  
`is_directory` (u115.Task attribute), 17  
`is_root` (u115.Directory attribute), 20  
`is_torrent` (u115.File attribute), 19  
`is_transferred` (u115.Task attribute), 17

### J

JobError (class in u115), 21

### L

`list()` (u115.Directory method), 20  
`list()` (u115.Task method), 17  
`load_cookies()` (u115.API method), 13  
`login()` (u115.API method), 13  
`logout()` (u115.API method), 14

### M

`max_entries_per_load` (u115.Directory attribute), 21  
`mkdir()` (u115.API method), 14  
`mkdir()` (u115.Directory method), 21  
`move()` (u115.API method), 14  
`move()` (u115.Directory method), 21  
`move()` (u115.File method), 19

### N

`num_tasks_per_page` (u115.API attribute), 14

### O

`open_torrent()` (u115.File method), 20

## P

parent (u115.Directory attribute), 21  
parent (u115.Task attribute), 17  
Passport (class in u115), 16  
post() (u115.RequestHandler method), 15  
proapi\_url (u115.API attribute), 14

## R

receiver\_directory (u115.API attribute), 14  
referer\_url (u115.API attribute), 14  
reload() (u115.Directory method), 21  
reload() (u115.File method), 20  
Request (class in u115), 15  
RequestFailure (class in u115), 21  
RequestHandler (class in u115), 15  
RequestsLWPCookieJar (class in u115), 16  
RequestsMozillaCookieJar (class in u115), 16  
Response (class in u115), 15  
root\_directory (u115.API attribute), 14

## S

save\_cookies() (u115.API method), 14  
search() (u115.API method), 14  
select() (u115.TorrentFile method), 18  
selected\_files (u115.Torrent attribute), 18  
send() (u115.RequestHandler method), 15  
status\_human (u115.Task attribute), 18  
submit() (u115.Torrent method), 18

## T

Task (class in u115), 16  
task\_count (u115.API attribute), 14  
task\_quota (u115.API attribute), 14  
TaskError (class in u115), 21  
Torrent (class in u115), 18  
TorrentFile (class in u115), 18  
torrents\_directory (u115.API attribute), 14

## U

unselect() (u115.TorrentFile method), 18  
unselected\_files (u115.Torrent attribute), 18  
upload() (u115.API method), 14  
url (u115.File attribute), 20  
user\_id (u115.API attribute), 15  
username (u115.API attribute), 15

## W

web\_api\_url (u115.API attribute), 15