# pyfilemail Documentation

*Release 0.5.1*

**Daniel Flehner Heen**

**Sep 07, 2017**

# Contents

Pyfilemail is a command line tool and API for sending and recieving files with Filemail based on requests and filemail's API.

To avoid nagging about API KEY you should register and get one here. If you register for a paid plan you unlock all features and will be able to add/delete/update your transfers/contacts/group/company settings. Without registering you'll still be able to send files as a free plan user but remember to use the `--free` argument in the command line tool.

For more info on the different plans please go to Filemail

I've tried to keep this api as simple as possible and rely on filemail's own validation of data to keep you all in check :) The reason for this is that I don't work at filemail and have no insight in what validation they have for the data passed. It also saves me a lot of head ache writing rock solid validation code and I think this is a more flexible way of doing it if the Filemail team decides to change the API in any way.

Appart from `pyfilemail.User` and `pyfilemail.Transfer` classes, all return objects from filemail are `dict` objects based on json responses.

So far this has been developed and tested on Ubuntu 16.04. I'll try to get my hands on a Windows and OSX machine and addapt the code to make sure it works there as well.

API documentation is available at readthedocs

Any feedback is more than welcome and please report bugs through github

CHAPTER 1

# Installation

```
pip install pyfilemail
```

CHAPTER 2

# Disclaimer

I use this software myself and have not yet experienced broken files, but I take no responsibility for the files sent or received using pyfilemail whether it comes to content or state of files passing through. Pyfilemail is still work in progress and parts of the API might change as I develop and test it further.

CHAPTER 3

# Command line example

```
pyfilemail \
--from myemail@somedomain.com \
--to lucky.b@receiver.com \
--free \
--subject "Amazing document!" \
--message "Have you seen how amazingly big this document is?" \
--payload /path/to/file.ext /path/to/folder/
```

CHAPTER 4

# Add API KEY

You can add the API KEY to the local config file with the `--add-api-key` argumet.

```
pyfilemail --add-api-key YOUR-API-KEY-FROM-FILEMAIL
```

# CHAPTER 5

## netrc

You may use a .netrc file to store login information. Make sure you restrict access to only allow your own user to read it.

```
#$HOME/.netrc example:
machine yourfilemailuser@email.com
        login yourfilemailuser@email.com
        password topsecretpassword
```

CHAPTER 6

# Command line help

```
usage: pyfilemail [-h] [--version] [--add-api-key ADD_API_KEY] [--free] [-nc]
              [--compress] [--confirm] [--quiet] [--days 3]
              [--downloads 0] [--message MESSAGE] [--notify]
              [--subject SUBJECT]
              [--to recipient@receiver.com [recipient@receiver.com ...]]
              [--password PASSWORD] [--from USERNAME] [--store-password]
              [--delete-password] [--payload PAYLOAD [PAYLOAD ...]]

Command line Filemail transfer through Python

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  --add-api-key ADD_API_KEY
                        Add API KEY from Filemail to local config file
  --free                Send files without a registered Filemail account
  -nc, --no-checksum    Skip calculating checksum on added files
  --compress            Compress (ZIP) data before sending?
  --confirm             Email confirmation after sending the files?
  --quiet               Log only warnings to console
  --days 3              Number of days the file(s) are available for download
  --downloads 0         Number of times the file(s) may be downloaded.
                        0=unlimited
  --message MESSAGE     Message to the recipient(s) of the transfer
  --notify              Notify when recipients download your files?
  --subject SUBJECT     Subject of email sent with transfer
  --to recipient@receiver.com [recipient@receiver.com ...]
                        Recipient(s) of the transfer (email addresses)
  --password PASSWORD   Protect transfer with the supplied password
  --from USERNAME       Your email address
  --store-password      Store user password in keychain if available
  --delete-password     Delete password stored in keychain
  --payload PAYLOAD [PAYLOAD ...]
                        File(s) and/or folder(s) to transfer
```

# CHAPTER 7

## Python API examples

```
import pyfilemail

# Setup a transfer

# Initialize Filemail with as free (as in free beer) user
user = pyfilemail.User(username='user@mailprovider.com')

transfer = pyfilemail.Transfer(user,
                               to='lucky@recipient.com',
                               subject='My BIG file no email can handle',
                               message='You will not belive the speed of this␣
↪download!',
                               notify=True,
                               confirmation=True,
                               days=7,
                               password='JuSt2BeSaf£')

# Add a single file to transfer queue
transfer.add_files('/path/to/my/BIG_file.ext')

# Add multiple files
list_of_files = ['/path/to/my/BIG_file_1.ext',
                 '/path/to/my/BIG_file_2.ext',
                 '/path/to/my/BIG_file_3.ext']

transfer.add_files(list_of_files)

# Send files to recipient(s)
transfer.send(auto_complete=True)

# Login to a registered Filemail account
user = pyfilemail.User(username='user@mailprovider.com',
                       password='YourSecretPassword2014')

# List all prior transfers
```

```
transfers = user.get_sent(expired=True)

# Get contacts
user.get_contacts()

# Get one single contact
contact = user.get_contact('contact@email.address.com')

# Update that contact
contact['name'] = 'Mr. Orange'
user.update_contact(contact)

# Delete contact
unfriendly = user.get_contact('contact@email.address.com')
user.delete_contact(unfriendly)

# Download received transfers for the past 7 days
transfers = user.get_received(age=7)
for transfer in transfers:
    transfer.download(destination='/home/myname/Downloads')

# Logout
user.logout()
```

# CHAPTER 8

# API

pyfilemail.**get_configfile**()
Return full path to configuration file.

- Linux: ~/.local/share/pyfilemail
- OSX: ~/Library/Application Support/pyfilemail
- Windows: C:\Users\{username}\AppData\Local\pyfilemail

**Return type** str

pyfilemail.**load_config**()
Load configuration file containing API KEY and other settings.

**Return type** str

pyfilemail.**login_required**(*f*)
Decorator function to check if user is loged in.

**Raises** FMBaseError if not logged in

pyfilemail.**save_config**(*config*)
Save configuration file to users data location.

- Linux: ~/.local/share/pyfilemail
- OSX: ~/Library/Application Support/pyfilemail
- Windows: C:\Users\{username}\AppData\Local\pyfilemail

**Return type** str

**class** pyfilemail.**User**(*username*, *password=None*)
This is the entry point to filemail. If you use a registered username you'll need to provide a password to login. If no password is passed during init a search for password is done in $HOME/.netrc You may also login at a later time with the *User.login()* function.

**Parameters**

- **username** (*str*) – your email/username
- **password** (*str*) – filemail password if registered username is used

```
#$HOME/.netrc example:
machine yourfilemailuser@email.com
        login yourfilemailuser@email.com
        password topsecretpassword
```

**add_contact**(**args*, ***kwargs*)
Add new contact.

**Parameters**

- **name** (str, unicode) – name of contact
- **email** (str, unicode) – email of contact

**Returns** contact information for new current user

**Return type** dict

**add_contact_to_group**(**args*, ***kwargs*)
Add contact to group

**Parameters**

- **contact** (str, unicode, dict) – name or contact object
- **group** (str, unicode, dict) – name or group object

**Return type** bool

**add_group**(**args*, ***kwargs*)
Add new contact group

**Parameters** **name** (str, unicode) – name of new group

**Return type** dict with group data

**company_add_user**(**args*, ***kwargs*)
Add a user to the company account.

**Parameters**

- **email** (str or unicode) –
- **name** (str or unicode) –
- **password** (str or unicode) – Pass without storing in plain text
- **receiver** (bool) – Can user receive files
- **admin** (bool) –

**Return type** bool

**delete_contact**(**args*, ***kwargs*)
Delete contact.

**Parameters** **contact** (dict) – with *contactid*

**Return type** bool

**delete_group**(**args*, ***kwargs*)
Delete contact group

**Parameters** **name** (str, unicode) – of group

**Return type** bool

**get_company_info**(**args*, ***kwargs*)
Get company settings from Filemail

**Return type** dict with company data

**get_company_user**(**args*, ***kwargs*)
Get company user based on email.

**Parameters** **email** (str, unicode) – address of contact

**Return type** dict with contact information

**get_company_users**(**args*, ***kwargs*)
Get company users from Filemail

**Return type** list of dict with user data

**get_contact**(**args*, ***kwargs*)
Get Filemail contact based on email.

**Parameters** **email** (str, unicode) – address of contact

**Return type** dict with contact information

**get_contacts**(**args*, ***kwargs*)

**Get contacts from Filemail. Usually people you've sent files** to in the past.

**Return type** list of dict objects containing contact information

**get_group**(**args*, ***kwargs*)
Get contact group by name

**Parameters** **name** (str, unicode) – name of group

**Return type** dict with group data

**get_groups**(**args*, ***kwargs*)
Get contact groups

**Return type** list of dict with group data

**get_received**(**args*, ***kwargs*)

**Retrieve a list of transfers sent to you or your company** from other people.

**Parameters**

- **age** (int) – between 1 and 90 days.
- **for_all** (bool) – If True will return received files for all users in the same business. (Available for business account members only).

**Return type** list of *Transfer* objects.

**get_sent**(**args*, ***kwargs*)
Retreve information on previously sent transfers.

**Parameters**

- **expired** (*bool*) – Whether or not to return expired transfers.
- **for_all** (*bool*) – Get transfers for all users. Requires a Filemail Business account.

**Return type** list of *pyfilemail.Transfer* objects

**get_user_info**(**args*, ***kwargs*)
Get user info and settings from Filemail.

**Parameters** **save_to_config** (bool) – Whether or not to save settings to config file

**Return type** dict containig user information and default settings.

**is_registered**
If user is a registered user or not.

**Return type** bool

**logged_in**
If registered user is logged in or not.

**Return type** bool

**login**(*password*)
Login to filemail as the current user.

**Parameters** **password** (str) –

**logout**(**args*, ***kwargs*)
Logout of filemail and closing the session.

**remove_contact_from_group**(**args*, ***kwargs*)
Remove contact from group

**Parameters**

- **contact** (str, unicode, dict) – name or contact object
- **group** (str, unicode, dict) – name or group object

**Return type** bool

**rename_group**(**args*, ***kwargs*)
Rename contact group

**Parameters**

- **group** (str, unicode, dict) – group data or name of group
- **newname** (str, unicode) – of group

**Return type** bool

**transfers_complete**
Check if all transfers are completed.

**update_company**(**args*, ***kwargs*)
Update company settings

**Parameters** **company** (dict) – updated settings

**Return type** bool

**update_company_user**(**args*, ***kwargs*)
Update a company users settings

**Parameters**

- **email** (str or unicode) – current email address of user
- **userdata** (dict) – updated settings

**Return type** bool

**update_contact**(**args*, ***kwargs*)
Update name and/or email for contact.

**Parameters** **contact** (dict) – with updated info

**Return type** bool

**update_user_info**(**args*, ***kwargs*)
Update user info and settings.

**Parameters** ****kwargs** – settings to be merged with User.get_configfile() setings and sent to Filemail.

**Return type** bool

**class** pyfilemail.**Transfer**(*fm_user*, *to=None*, *subject=None*, *message=None*, *notify=False*, *confirmation=False*, *days=3*, *downloads=0*, *password=None*, *checksum=True*, *zip_=False*, *_restore=False*)
This is is the gateway to sending and recieving files through filemail.

**Parameters**

- **fm_user** (*pyfilemail.User*, str) – username
- **to** (*str, list*) – recipient(s)
- **subject** (*str, unicode*) –
- **message** (*str, unicode*) –
- **notify** (*bool*) – Notify when recipient(s) download files
- **confirmation** (*bool*) – Receive confirmation email when files are uploaded
- **days** (*int*) – Number of days files are available for download
- **downloads** – Number of times files may be downloaded
- **password** (*str, unicode*) – Protect download with given password
- **checksum** (*bool*) – Create checksum of added files (a bit slower process)
- **zip** (*bool*) – Compress files in a zip file before sending

**add_files**(*files*)
Add files and/or folders to transfer. If *Transfer.compress* attribute is set to True, files will get packed into a zip file before sending.

**Parameters** **files** (*str, list*) – Files or folders to send

**cancel**()
Cancel the current transfer.

**Return type** bool

**complete**()
Completes the transfer and shoots off email(s) to recipient(s).

**compress**(**args*, ***kwargs*)

**Compress files on the server side after transfer complete** and make zip available for download.

**Return type** bool

**delete**(**args*, ***kwargs*)
Delete the current transfer.

**Return type** bool

**delete_file**(**args*, ***kwargs*)
Delete file from transfer.

**Parameters** **fmfile** (dict) – file data from filemail containing fileid

**Return type** bool

**download**(*files=None*, *destination=None*, *overwrite=False*, *callback=None*)
Download file or files.

**Parameters**

- **files** (list of dict with file data from filemail) – file or files to download
- **destination** (str or unicode) – destination path (defaults to users home directory)
- **overwrite** (bool) – replace existing files?
- **callback** (func) – callback function that will receive total file size and written bytes as arguments

**files**

**Returns** List of files/folders added to transfer

**Return type** list

**forward**(*to*)
Forward prior transfer to new recipient(s).

**Parameters** **to** (list or str or unicode) – new recipients to a previous transfer. Use list or comma seperatde str or unicode list

**Return type** bool

**get_file_specs**(*filepath*, *keep_folders=False*)
Gather information on files needed for valid transfer.

**Parameters**

- **filepath** (*str, unicode*) – Path to file in question
- **keep_folders** (*bool*) – Whether or not to maintain folder structure

**Return type** dict

**get_files**()
Get information on file in transfer from Filemail.

**Return type** list of dict objects with info on files

**is_complete**

**Return type** bool True if transfer is complete

**logged_in**
If registered user is logged in or not.

**Return type** bool

**rename_file**(**args*, ***kwargs*)
Rename file in transfer.

**Parameters**

- **fmfile** (dict) – file data from filemail containing fileid
- **newname** (str or unicode) – new file name

**Return type** bool

**send**(**args*, ***kwargs*)
Begin uploading file(s) and sending email(s). If *auto_complete* is set to False you will have to call the *Transfer.complete()* function at a later stage.

**Parameters**

- **auto_complete** (bool) – Whether or not to mark transfer as complete and send emails to recipient(s)
- **callback** (func) – Callback function which will receive total file size and bytes read as arguments

**share**(**args*, ***kwargs*)
Share transfer with new message to new people.

**Parameters**

- **to** (list or str or unicode) – receiver(s)
- **sender** (str or unicode) – Alternate email address as sender
- **message** (str or unicode) – Meggase to new recipients

**Rtyep** bool

**transfer_id**
Get the transfer id for the current Transfer.

**Return type** unicode with transfer id

**update**(**args*, ***kwargs*)
Update properties for a transfer.

**Parameters**

- **message** (str or unicode) – updated message to recipient(s)
- **subject** (str or unicode) – updated subject for trasfer
- **days** (int) – updated amount of days transfer is available
- **downloads** (int) – update amount of downloads allowed for transfer
- **notify** (bool) – update whether to notifiy on downloads or not

**Return type** bool

CHAPTER 9

# Indices and tables

- genindex
- search

# Python Module Index

## p

# Index