# dmglib Documentation

*Release 0.9.5*

**Jakob Rieck**

**Mar 20, 2024**

# CONTENTS:

*dmglib* is a small Python wrapper library for the *hdiutil* command on macOS. Its only job is to make it easy and painless to deal with macOS disk image (DMG) files from Python.

CHAPTER

# ONE

# API DOCUMENTATION

The most straight-forward way to use the functionality this package provides is simply to use a context manager:

dmglib.**attachedDiskImage**(*path: str*, *keyphrase=None*)

Context manager to work with a disk image.

The context manager returns the list of mount points of the attached volumes. There is always at least one mount point available, otherwise attaching fails. The caller needs to catch exceptions (see documentation for the *DiskImage* class), or call the appropriate methods beforehand (*dmg_is_encrypted()*, ...).

Example:

```
with dmg.attachedDiskImage('path/to/disk_image.dmg',
                           keyphrase='sample') as mount_points:
    print(mount_points)
```

Apart from the context manager, you may also explicitly use the *DiskImage* class:

**class** dmglib.**DiskImage**(*path*, *keyphrase=None*)

Class representing macOS Disk Images (.dmg) files.

Initialize a disk image object. Note: Simply constructing the object does not attach the DMG. Use the *DiskImage.attach()* method for that.

**Parameters**

- **path** – The path to the disk image
- **keyphrase** – Optional argument for password protected images

**Raises**

- ***AlreadyAttached*** – The disk image is already attached on the system.
- ***InvalidDiskImage*** – The disk image is not a valid disk image.
- ***PasswordRequired*** – A password is required but none was provided.
- ***PasswordIncorrect*** – A incorrect password was supplied.

**attach**(*mountpoint=None*)

Attaches a disk image.

**Parameters**
**mountpoint** – Optional path where disk image should be mounted.

**Returns**
List of mount points.

**Raises**

- *InvalidOperation* – This disk image has already been attached.
- *LicenseAgreementNeedsAccepting* – The image cannot be automatically mounted due to a license agreement.
- *AttachingFailed* – Could not attach the disk image or no volumes on mounted disk.

**convert**(*path: str*, *disk_format:* DiskFormat) → str

Converts a disk image to a different format.

**Parameters**

- **path** – The path where to store the converted disk image.
- **disk_format** – One of the hdiutil supported disk image formats, see *DiskFormat*

**Returns**

The filepath where the converted disk image was stored. Note that this may differ from *path* in case the correct file extension for the chosen disk format differs from the file extension provided as part of *path*.

**Raises**

*ConversionFailed* – hdiutil could not convert the disk image to the specified format.

**detach**(*force=True*)

Detaches a disk image.

**Parameters**

**force** – ignore open files on mounted volumes. See *man 1 hdiutil*.

**Raises**

- *InvalidOperation* – The disk image was not attached on the system.
- *DetachingFailed* – Detaching failed for unknown reasons.

**has_license_agreement**() → bool

Checks whether the disk image has an attached license agreement.

DMGs with license agreements cannot be attached using this package.

## 1.1 Standalone functions

dmglib.**dmg_is_valid**(*path: str*) → bool

Checks the validity of the supplied disk image.

A disk image is valid according to this logic, if it is either not encrypted and valid according to hdiutil, or encrypted according to hdiutil.

dmglib.**attached_images**() → list

Obtain a list of paths to disk images that are currently attached.

dmglib.**dmg_already_attached**(*path: str*) → bool

Checks whether the disk image at the supplied path has already been attached.

Querying the system for further information about already attached images fails with a resource exhaustion error message.

dmglib.**dmg_is_encrypted**(*path: str*) → bool

Checks whether DMG at the supplied path is password protected.

dmglib.**dmg_check_keyphrase**(*path: str*, *keyphrase: str*) → bool

Checks the keyphrase for the disk image at the supplied path.

**Note:** This function assumes the DiskImage is encrypted and raises an exception if it is not.

**Parameters**

- **path** – path to disk image for which to check the keyphrase
- **keyphrase** – keyphrase to check

**Raises**

*InvalidOperation* – the disk image was not encrypted.

dmglib.**dmg_get_mountpoints**(*path: str*) → dict

Returns mountpoints of the already attached dmg.

**Parameters**

**path** – path to the already attached disk image.

**Returns**

Dict with mountpoints.

**Raises**

*InvalidOperation* – if image is not already attached.

dmglib.**dmg_detach_already_attached**(*path: str*, *force=True*)

Detaches a disk image without DiskImage object, e.g. for creating it.

**Parameters**

- **path** – path to the disk image
- **force** – ignore open files on mounted volumes. See *man 1 hdiutil*.

**Raises**

- *InvalidOperation* – The disk image was not attached on the system.
- *DetachingFailed* – Detaching failed for unknown reasons.

dmglib.**dmg_create_blank**(*path: str*, *disk_type:* DiskCreateBlankFormat = *None*, *fs_type:* FsFormat = *None*, *size=None*, *rename_sparse=False*)

Creates blank DMG. Note: Doesn't construct the DiskImage object.

**Parameters**

- **path** – The path to the disk image
- **disk_type** – Optional argument, specifies disk type for blank images
- **fs_type** – Optional argument, specifies filesystem type for blank images
- **size** – Optional argument, specifies size for blank images
- **rename_sparse** – Optional argument, if true renames '.dmg.sparseimage' to '.dmg', for sparseimages

**Raises**

*CreatingFailed* – Error while creating blank disk images

## 1.2 Exceptions

**exception** dmglib.**InvalidDiskImage**

The disk image is deemed invalid and therefore cannot be attached.

**exception** dmglib.**InvalidOperation**

An invalid operation was performed by the user.

Examples include trying to detach a dmg that was never attached or trying to attach a disk image twice.

**exception** dmglib.**ConversionFailed**

Error to indicate that conversion failed

**exception** dmglib.**AttachingFailed**

Attaching failed for unknown reasons.

**exception** dmglib.**DetachingFailed**

Error to indicate a volume could not be detached successfully.

**exception** dmglib.**AlreadyAttached**

The disk image has already been attached previously.

**exception** dmglib.**PasswordRequired**

No password was required even though one was required.

**exception** dmglib.**PasswordIncorrect**

An incorrect password was supplied for the disk image.

**exception** dmglib.**LicenseAgreementNeedsAccepting**

Error indicating that a license agreement needs accepting.

**exception** dmglib.**CreatingFailed**

Error to indicate an image could not be created successfully.

## 1.3 Enumerations

**class** dmglib.**DiskFormat**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Supported disk image formats for convert verb.

**COMPRESSED = 'UDZO'**

**COMPRESSED_ADC = 'UDCO'**

**COMPRESSED_BZIP2 = 'UDBZ'**

**COMPRESSED_LZFSE = 'UDFO'**

**COMPRESSED_LZMA = 'ULMO'**

**DISK_COPY = 'DC42'**

**ENTIRE_DEVICE = 'UFBI'**

**IPOD_IMAGE = 'IPOD'**

NDIF_COMPRESSED = 'ROCo'

NDIF_KEN_CODE = 'Rken'

NDIF_READ_ONLY = 'Rdxx'

NDIF_READ_WRITE = 'RdWr'

OPTICAL_MASTER = 'UDTO'

READ_ONLY = 'UDRO'

READ_WRITE = 'UDRW'

SPARSE = 'UDSP'

SPARSE_BUNDLE = 'UDSB'

UDIF_STUB = 'UDxx'

class dmglib.**DiskCreateBlankFormat**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Supported disk image formats for create verb (blank images).

OPTICAL_MASTER = 'UDTO'

READ_WRITE_IMAGE = 'UDIF'

SPARSEBUNDLE_IMAGE = 'SPARSEBUNDLE'

SPARSE_IMAGE = 'SPARSE'

class dmglib.**FsFormat**(*value*, *names=None*, **values*, *module=None*, *qualname=None*, *type=None*, *start=1*, *boundary=None*)

Supported filesystem formats.

APFS = 'APFS'

APFS_CASE = 'Case-sensitive APFS'

EXFAT = 'ExFAT'

MAC_OS_EXTENDED = 'HFS+'

MAC_OS_EXTENDED_CASE = 'Case-sensitive HFS+'

MAC_OS_EXTENDED_CASE_JOUR = 'Case-sensitive Journaled HFS+'

MAC_OS_EXTENDED_JOUR = 'Journaled HFS+'

MS_DOS_FAT = 'MS-DOS'

MS_DOS_FAT12 = 'MS-DOS FAT12'

MS_DOS_FAT16 = 'MS-DOS FAT16'

MS_DOS_FAT32 = 'MS-DOS FAT32'

UNIVERSAL_DISK = 'UDF'

CHAPTER

# TWO

# EXAMPLE

The following example program attempts to mount the supplied disk image and iterates over the files in the root directory of all its mount points. Note that error handling has largely been ignored to keep the example as simple as possible:

```
import dmglib
import sys
import os


def main():
    if len(sys.argv) <= 1:
        print("Usage: program dmgpath")
        return

    dmgpath = sys.argv[1]
    dmg = dmglib.DiskImage(dmgpath)

    if dmg.has_license_agreement():
        print("Cannot attach disk image.")
        return

    for mount_point in dmg.attach():
        for entry in os.listdir(mount_point):
            print('{} -- {}'.format(mount_point, entry))

    dmg.detach()


if __name__ == '__main__':
    main()
```

CHAPTER

THREE

# INDICES AND TABLES

- genindex
- modindex
- search

# INDEX