
imagemounter Documentation

Release 1.5.1

Ralph Broenink, Peter Wagenaar

Apr 04, 2022

Contents

1	Contents	3
1.1	Installation	3
1.2	Dependencies	4
1.3	Command-line usage	5
1.4	Python interface	8
1.5	File and volume system specifics	9
1.6	Release notes	14
	Python Module Index	25
	Index	27

imagemounter is a command-line utility and Python package to ease the mounting and unmounting of EnCase, Affuse, vmdk and dd disk images (and other formats supported by supported tools). It supports mounting disk images using xmount (with optional RW cache), affuse, ewfmount, vmware-mount and qemu-nbd; detecting DOS, BSD, Sun, Mac and GPT volume systems; mounting FAT, Ext, XFS UFS, HFS+, LUKS and NTFS volumes, in addition to some less known filesystems; detecting (nested) LVM volume systems and mounting its subvolumes; and reconstructing Linux Software RAID arrays.

In its default mode, imagemounter will try to start mounting the base image on a temporary mount point, detect the volume system and then mount each volume separately. If it fails finding a volume system, it will try to mount the entire image as a whole if it succeeds in detecting what it actually is.

Note: Not all combinations of file and volume systems have been tested. If you encounter an issue, please try to change some of your arguments first, before creating a new GitHub issue.

Warning: Mounting disks and volumes from unknown sources may pose an important security risk (especially since you probably need to run imagemounter as root).

1.1 Installation

If you need an installation with basic support, you are suggested to run the following commands:

```
apt-get install xmount ewf-tools afflib-tools sleuthkit  
pip3 install imagemounter  
imount --check
```

The latter command will list all other packages you could install to expand the capabilities of imagemounter.

1.1.1 Python packages

This package does not require other packages, though the `termcolor` package is recommended if you are using the **imount** command line utility with the `--color` argument.

If you wish to use `pytsk3` support, you require *python-dev* and *libtsk-dev*. For compilation, the *build-essential* package from your distribution is also required. After that, you can easily install the `pytsk3` package from PyPI (**pip** requires the `--pre` flag to allow installing the package).

1.1.2 Other dependencies

This package highly depends on other utilities to be present on your system. For a full installation, you require more tools. You can run `imount --check` to get a full list of all required tools.

A basic installation contains at least one of the mount tools. Highly recommended is also `fsstat`, others are required for specific file system types.

You can install `vmware-mount` by installing VMware Workstation on your system.

1.2 Dependencies

The `imgemounter.dependencies` module defines several optional and required dependencies to use `imgemounter`. This data is used by the `imount --check` command, and can also be accessed via Python code:

```
from imgemounter.dependencies import ewfmount

...

if ewfmount.is_available:
    do_something_with_ewfmount()
else:
    print(ewfmount.printable_status)
```

Full list of dependencies:

- `imgemounter.dependencies.affuse`
- `imgemounter.dependencies.bdemount`
- `imgemounter.dependencies.blkid`
- `imgemounter.dependencies.cryptsetup`
- `imgemounter.dependencies.disktype`
- `imgemounter.dependencies.ewfmount`
- `imgemounter.dependencies.file`
- `imgemounter.dependencies.fsstat`
- `imgemounter.dependencies.lvm`
- `imgemounter.dependencies.magic`
- `imgemounter.dependencies.mdadm`
- `imgemounter.dependencies.mmls`
- `imgemounter.dependencies.mount_jffs2`
- `imgemounter.dependencies.mount_ntfs`
- `imgemounter.dependencies.mount_squashfs`
- `imgemounter.dependencies.mount_xfs`
- `imgemounter.dependencies.mountavfs`
- `imgemounter.dependencies.parted`
- `imgemounter.dependencies.pytsk3`
- `imgemounter.dependencies.qemu_nbd`
- `imgemounter.dependencies.vmfs_fuse`
- `imgemounter.dependencies.vmware_mount`
- `imgemounter.dependencies.vshadowmount`
- `imgemounter.dependencies.xmount`

1.2.1 API

The following classes are how dependencies are represented within imagemounter:

1.3 Command-line usage

One of the core functionalities of *imagemounter* is the command-line utility **imount** that eases the mounting and unmounting of different types of disks and volumes. In its most basic form, the utility accepts a positional argument pointing to a disk image, disk or volume, e.g.:

```
imount disk.E01
```

Multiple files can be passed to this command, allowing the mounting of volume systems that span multiple disks, which can be useful for those wishing to reconstruct a system that entailed multiple disks or for reconstructing RAID arrays.

By default, **imount** will mount each single volume in `/tmp` and wait until you confirm an unmount operation. Common usage is therefore to keep **imount** running in a separate window and perform other operations in a second window.

1.3.1 Arguments

The **imount** utility requires one (or more) positional arguments and offers the ability to pass several optional arguments.

<image> [**<image>** ...]

The positional argument(s) should provide the path(s) to the disk images you want to mount. Many different formats are supported, including the EnCase evidence format, split dd files, mounted hard drives, etc. In the case of split files, you can refer to the folder containing these files.

If you specify more than one file, all files are considered to be part of the same originating system, which is relevant for the `--reconstruct` command-line option.

Arguments that immediately exit

Some useful facilities.

--help

-h

Shows a help message and exits.

--version

Shows the current version and exits.

--check

Shows which third-party utilities you have installed for a correct functioning of imagemounter.

--unmount

-u

Option that will try to identify leftover files from previous **imount** executions and try to delete these. This will, for instance, clean leftover `/tmp/im_...` mounts and mountpoints. This command will allow you to review the actions that will be taken before they are done.

Can be combined with `--casename`, `--mountdir` and `--pretty` to specify which mount points to delete.

CLI behaviour

The next four command-line options alter the behaviour of the **imount** utility, but does not affect the behaviour of the underlying *imagemounter* module.

--wait

-w

Pauses the execution of the program on all warnings.

--keep

-k

Skips the unmounting at the end of the program.

--no-interaction

Never ask for input from the user, implies *--keep*.

--only-mount

Comma-separated list of volume indexes you want to mount. Other volumes are skipped.

--skip

Comma-separated list of volume indexes you do not want to mount.

--verbose

-v

Show verbose output. Repeat for more verbosity (up to 4).

--color

--no-color

Force toggle colorizing the output. Verbose message will be colored blue, for instance. Requires the *termcolor* package.

Additional features

This command-line option enables an additional and useful feature.

--reconstruct

-r

Attempts to reconstruct the full filesystem tree by identifying the last mountpoint of each identified volume and bindmounting this in the previous root directory. For instance, if volumes have previously been mounted at */* , */var* and */home* ; */var* and */home* will be bind-mounted in */* , providing you with a single filesystem tree in the mount location of */* that is easily traversible.

This only works with Linux-based filesystems and only if */* can be identified.

Implies *--stats*.

--carve

Carves the filesystem for missing files.

--vshadow

Also mounts volume shadow copies

Mount behaviour

These arguments alter some pieces of the mount behaviour of *imagemounter*, mostly to ease your work.

--mountdir <directory>

-md <directory>

Specifies the directory to place volume mounts. Defaults to a temporary directory.

--pretty

-p

Uses pretty names for volume mount points. This is useful in combination with `--mountdir`, but you should be careful using this option. It does not provide a fallback when the mount point is not available or other issues arise. It can also not be cleaned with `--clean`.

--casename

-cn

Use to specify the case name, which is used in pretty mounts, but also for the location of the mountdir. Useful if you want to be able to identify the mountpoints later.

--read-write

-rw

Will use read-write mounts. Written data will be stored using a local write cache.

Implies `--method xmount`.

Advanced options

While *imagemounter* will try to automatically detect as much as possible, there are some cases where you may wish to override the automatically detected options. You can specify which detection methods should be used and override the volume system and file system types if needed.

--disk-mounter <method>

-m <method>

Specifies the method to use to mount the base image(s). Defaults to automatic detection, though different methods deliver different results. Available options are *xmount*, *affuse* and *ewfmount* (defaulting to *auto*).

If you provide *dummy*, the base is not mounted but used directly.

--volume-detector <method>

-d <method>

Specifies the volume detection method. Available options are *pytsk3*, *mmls*, *parted* and *auto*, which is the default. Though *pytsk3* and *mmls* should in principle deliver identical results, *pytsk3* can be considered more reliable as this uses the C API of The Sleuth Kit (TSK). However, it also requires *pytsk3* to be installed, which is not possible with Py3K.

--vstypes <types>

Specifies the type of the volume system, defaulting to *detect*. However, detection may not always succeed and valid options are *dos*, *bsd*, *sun*, *mac*, *gpt* and *dbfller*, though the exact available options depend on the detection method and installed modules on the operating system.

--fstypes <types>

Specifies the filesystem of a volume to use. Available options include *ext*, *ufs*, *ntfs*, *luks*, *lvm* and *unknown*, with the latter simply mounting the volume without specifying type. See the command-line help for all available volume types.

Filesystem types are specified for each volume separately. You can use subvolumes, examples including:

```
1=ntfs
2=luks,2.0=lvm,2.0.1=ext
```

If you wish to specify a fallback to use if automatic detection fails, you can use the special question mark (?) volume index. If you wish to override automatic detection at all for all unspecified volumes, you can use the asterisk (*) volume type. There is no point in specifying both a question mark and an asterisk.

--keys <keys>

Allows the specification of key information for each volume separately. This is similar to `--fstypes`, except

that you can only specify one key per argument (i.e. a comma is not interpreted as special). The format of the specific value depends on the volume type:

For BDE, you can use a single letter, followed by a colon, followed by the value. This leads to the following accepted formats, similar to how the **bde** command interprets input:

```
k:full volume encryption and tweak key
p:passphrase
r:recovery password
s:file to startup key (.bek)
```

For LUKS, you can use a similar format:

```
p:passphrase
f:key-file
m:master-key-file
```

--lazy-unmount

Enables to unmount the volumes and disk lazily when the direct unmounting of the volumes fails.

Advanced toggles

imount has some facilities that automatically detect some types of disks and volumes. However, these facilities may sometimes fail and can be disabled if needed.

--single

--no-single

imount will, by default, try to detect whether the disk that is being mounted, contains an entire volume system, or only a single volume. If you know your volumes are not single volumes, or you know they are, use *--no-single* and *--single* respectively.

Where *--single* forces the mounting of the disk as a single volume, *--no-single* will prevent the identification of the disk as a single volume if no volume system is found.

1.4 Python interface

While **imount** heavily utilizes the Python API of *imagemounter*, this API is also available for other classes.

1.4.1 Data structure

The basic structure of *imagemounter* is the `imagemounter.ImageParser` class, which provides access to underlying `imagemounter.Disk` and `imagemounter.Volume` objects. Each file name passed to a new `imagemounter.ImageParser` object results in one `imagemounter.Disk` object. `imagemounter.Volume` objects are created by analysis of the `Disk` object (each volume generates one object, even if it is not mountable), and each `imagemounter.Volume` can have one or more subvolumes.

For instance, a LUKS volume may contain a LVM system that contains a Ext volume. This would create a `Disk` with a `Volume` containing a `Volume` which contains the actual Ext `Volume`. Subvolumes are managed through `imagemounter.VolumeSystem`'s, which is used by both the `:class:`Volume` and `Disk` classes.

Most operations are managed on a `Volume` level, although individual disk file mounting (and volume detection) is performed on a `Disk` level and reconstruction is performed on a `ImageParser` level. This means the following main parts make up the Python package:

- `imagemounter.ImageParser`, maintaining a list of Disks, providing several methods that are carried out on all disks (e.g. `mount`) and `reconstruct`.
- `imagemounter.Disk`, which represents a single disk image and can be mounted, and maintain volumes. It is also responsible for maintaining the write cache. Although a `Disk` is able to detect volumes, a `Volume` has similar capabilities.
- `imagemounter.Volume`, which can detect its own type and fill its stats, can be mounted, and maintain subvolumes.
- `imagemounter.VolumeSystem`, which is used to manage subvolumes and can detect volumes from a volume system.

All three classes maintain an `init()` method that yields the volumes below it. You should call `clean()` on the parser as soon as you are done; you may also call `unmount()` on separate volumes or disks, which will also unmount all volumes below it. Warning: unmounting one of the RAID volumes in a RAID array, causes the entire array to be unmounted.

1.4.2 Reference

If you utilize the API, you typically only require the `ImageParser` object, e.g.:

```
parser = ImageParser(['/path/to/disk'])
for v in parser.init():
    print v.size
root = parser.reconstruct()
print root.mountpoint
parser.clean()
```

The best example of the use of the Python interface is the **`imount`** command. The entirety of all methods and attributes is documented below.

ImageParser

Disk

Volume

VolumeSystem

Unmounter

1.5 File and volume system specifics

This section contains specifics on different file systems and volume systems supported by `imagemounter`. This section is not complete and does not cover every edge case. You are invited to amend this section with additional details.

1.5.1 File systems

ext

`ext2`, `ext3` and `ext4` are all supported by `imagemounter`. All mounts use the `ext4` drivers, allowing us to specify the `noload` argument to the mount subsystem.

UFS

Supported is UFS2, as we explicitly pass in the UFS2 driver type in the mount call. This may result in some UFS volumes not mounting. There is currently no workaround for that.

Normally, a volume can be detected as both a UFS volume, as well as a BSD partition table. When the former happens, it may appear as if there is only a single volume, as this hides the other UFS volumes. Although detections have been amended for this, it is important to note that you can explicitly override detection using `--fstypes` if detection fails for some reason.

(See BSD volume system below for more information.)

Depending on your OS, you may need to run `modprobe ufs` to enable UFS support in your kernel.

NTFS

For mounting NTFS, you may need the *ntfs-3g* package. The `show_sys_files` option is enabled by default.

This file system type may (accidentally) be detected when a BDE volume should be used instead.

HFS / HFS+

No additional details.

LUKS

For mounting LUKS volumes, the **cryptsetup** command is used. At this point, it is not possible to specify more options to the cryptsetup subsystem. To specify keys, use `--keys`. The following values are accepted:

```
p:passphrase
f:key-file
m:master-key-file
```

Note that you can't provide multiple keys using a single `--keys` argument. Repeat the argument to accomplish this, e.g. `--keys 0=p:passphrase --keys 1=p:passphrase`.

To determine whether a volume is a LUKS volume, `cryptsetup isLuks` is called. This method should return true; if it doesn't, *imagemounter* will also not be able to mount the volume. The next step is to create a loopback device that is used to call `cryptsetup luksOpen <device> <name>`, where `name` is of the form `image_mounter_luks_<number>`. Additional details of the volume are extracted by using `cryptsetup status`. The actual `dd` image of the volume is mounted in `/dev/mapper/<name>` by the OS.

The LUKS volume will get a subvolume at index 0 with the file system description `LUKS Volume`. When this volume is a LVM volume that is not properly recognized by *imagemounter*, you could use something like the following to amend this:

```
imount image.E01 --fstypes=1=luks,1.0=lvm,1.0.0=ext --keys=1=p:passphrase
```

LUKS volumes are automatically unmounted by ending the script normally, but can't be unmounted by `--unmount`.

The LUKS volume type may not be automatically detected in some cases.

BDE (Bitlocker)

Bitlocker Drive Encrypted volumes are mounted using the **bdemount** command. **imagemounter** allows you to provide the crypto material to the command using `--keys`. Examples of valid commands are:

```
imount image.E01 --fstypes=2=bde --keys=2=p:passphrase
imount image.E01 --fstypes=2=bde --keys=2=r:recovery_password
```

See the manpage for **bdemount** for all valid arguments that can be passed to the subsystem (crypto material is provided by replacing `<key>:<value>` with `-<key> <value>`).

The BDE volume will get a subvolume at index 0 with the file system description `BDE Volume`. **imagemounter** should normally correctly detect this subvolume to be a NTFS volume.

BDE volumes are automatically unmounted by ending the script normally, but in some cases may not be properly unmounted by `--unmount`.

The BDE volume type may not be properly recognized and may instead be recognized as NTFS volume. You can override this by explicitly stating the volume type as in the examples above.

LVM

LVM systems host multiple volumes inside a single volume. **imagemounter** is able to detect these volumes on most occasions, though it may not always be possible to detect the file system type of the volumes inside the LVM.

Mounting an LVM is done by mounting the volume to a loopback device and running `lvm pvscan`. This should return a list of all LVMs on the system, but by matching the mount point of the base image, the script should be able to identify the volume group name. This name is then used to enable the LVM by running `vgchange -a y <name>`. Using `lvdisplay <name>`, the volumes inside the volume group are extracted. The volume themselves are found at the LV Path provided by this command.

Volumes inside a LVM are given the FS description `Logical Volume`. The file system types should be recognized properly by the detection methods, and otherwise `unknown` should work, but otherwise you could explicitly specify the file system type as follows:

```
imount image.E01 --fstypes=1=lvm,1.0=ext
```

Please note that many Linux based operating systems will try to mount LVMs for you. Although **imagemounter** tries to circumvent this automation, if you are unable to properly unmount, you should try to unmount through the interface of your OS first. Another useful command is `vgchange -a n` to disable all LVMs currently active (only use if you are not using a LVM for your own OS!).

Unmounting LVMs is supported both by properly closing from the script as well as by using `--unmount`

Linux Software RAID

Linux RAID volume support is provided by the `mdadm` command. A volume is added to a RAID array incrementally; the `mdadm` command is responsible for adding the volume to the correct array. The location of the RAID array is captured by **imagemounter** so it can be unmounted again. A subvolume will be added with the description `RAID volume` at index 0.

If the RAID volume can not be started directly after adding the volume, mounting will have succeeded, but the mountpoint will not be available yet. When another volume is added to the same RAID array, it will get the same (identical) subvolume as the original RAID volume. You should not mount it again. `init` will take care of both cases for you.

Warning: If, for any reason, you have multiple RAID volumes in the same RAID array, unmounting one of the volumes will also immediately unmount all other RAID volumes in the same array. Because of this, you should ensure that you keep all RAID volumes mounted until you are done building and examining a specific array.

RAID volumes are sometimes correctly detected, but there are also cases where the volume appears to *successfully* mount as another volume type. You should be very careful with this.

Note: A disk leveraging full disk RAID can be mounted as a single volume with the RAID filesystem type.

XFS

XFS is supported through the *xfspgrog*s package.

ISO (ISO9660)

No additional details.

UDF

No additional details.

FAT

FAT volumes, independent of type, are mounted through the VFAT driver.

exFAT

exFAT volumes are mounted by the exFAT driver. Note that exFAT volumes are sometimes recognized as NTFS volumes.

Another quirk may be that parted recognizes a single exFAT volume as a DOS partition table with some free space (also see [this comment](#)). Use another detection method or an explicit `--single` to amend this.

VMFS

VMFS is supported through the *vmfs-tools* package. Mounting is performed by finding a loopback device and using the `vmfs-fuse` command to mount this loopback on the mountpoint.

SquashFS

SquashFS is supported through the *squashfs-tools* package.

JFFS2

JFFS2 is supported through the *mtt-tools* package. JFFS2 is sometimes used by BIOS images and the like.

The following commands are executed to open a JFFS2 image, where `<size>` is given a buffer of 1.2 times the size of the volume:

```
modprobe -v mtd
modprobe -v jffs2
modprobe -v mtdram total_size=<size> erase_size=256
modprobe -v mtdblock
dd if=<path> of=/dev/mtd0
mount -t jffs2 /dev/mtdblock0 <mountpoint>
```

Warning: This filesystem type may not work while mounting multiple images of the same type at the same time.

Unmounting for this filesystem type is not fully supported.

CramFS

No additional details.

Minix

No additional details.

Dir

The `dir` filesystem type is not an actual mount type, but is used by *imagemounter* to indicate directories. This can be used in conjunction with the `AVFS` mount method, but basically just symlinks a directory to the mount location. It is provided for abstraction purposes.

Unknown

The `unknown` filesystem type is not an actual mount type, but used by *imagemounter* to indicate that the volume should be mounted without specifying the volume type. This is less specific and does not work in most cases (since it lacks the ability to provide additional options to the mount subsystem) but may result in the volume actually being able to be used.

The `unknown` filesystem type is used as fallback by default, and is for instance used if no specific volume type is provided by any of the detection methods other than 'Linux'. If you wish to override this default, and choose skipping mounting instead, you can also use the `none` filesystem type:

```
imount image.dd --fstypes=?=none
```

1.5.2 Volume systems

DOS (MBR)

In some cases, the DOS volume system is recognized as either a DOS or a GPT volume system. This appears to be a bug in The Sleuth Kit used by some detection methods. `imagemounter` works around this by choosing in this case for the GPT volume system and will log a warning. In the case that this is not the right choice, you must use `--vstype` to explicitly provide the correct volume system.

In the case you have picked the wrong volume system, you can easily spot this. If you see `GPT Safety Partition` popping up, you should have chosen GPT.

GPT

See the DOS/MBR volume system.

BSD

The BSD volume system (BSD disklabel) is commonly used in conjunction with UFS.

BSD volume `c` (BSD disk label uses letters to indicate the volumes, `imagemounter` will number this as volume 3) may appear to contain the entire volume set, and have the same offset as UFS volume `a`. The correct volume is volume `a`, and you should skip volume `c`. This is currently not fixed by `imagemounter`.

Sun

No additional details.

MAC

No additional details.

Detect

Lets the subsystem automatically decide the correct volume system type.

1.6 Release notes

We try to reduce backwards compatibility breakage only to major version releases, i.e. `X.0.0`. Minor releases (`1.X.0`) may include new features, whereas patch releases (`1.0.X`) will generally be used to fix bugs. Not all versions have followed these simple rules to date (and sometimes new features may creep up in patch releases), but we try to adhere them as much as possible :).

1.6.1 Release history

4.0.0 (unreleased)

This release contains some major improvements in the API of `imagemounter`, making large if-statements obsolete and obtaining more ‘Pythonic’ code where possible. It is a continuation of the efforts in version 3.0 and includes fixes from many contributors.

New features:

- Drop support for Python 2, bump version requirement for Python to 3.6.
- Allowed programmatic access to dependency checks (contributed by gtback). This enables you to write code that checks whether certain dependencies are satisfied.
- Changed the way file system types work. Large if-statements in the code have been replaced by class-based filesystem types. **Backwards incompatible** * The `fstype` attribute has been replaced with the `filesystem` attribute, but a backwards-compatible `attribute` is still available.
 - The `_paths` attribute is no longer available, but replaced by attributes on the `:cls:'FileSystem'` classes
 - The class-based filesystems implement a form of heuristic to determine the filesystem type.

Note: This change may result in file systems that were previously correctly detected suddenly not being identified correctly. If you encounter such an error, please submit a bug report stating the outputs of the different identifiers, the actual result and the expected result. You can obtain this information using `-vvvv` on the command line.

- carve and volume shadow copies are now also subvolumes.
- Add support for VBox disk images (vdi) (contributed by ruzzle)
- Add support for VHD volumes (contributed by Jarmo van Lenthe)

Bugfixes:

- Renamed `nbd` to `qemu-nbd` (contributed by Jarmo van Lenthe)
- Support for segmented filenames higher than 999 (contributed by jdosssett)
- Fixed `xmount` command order (contributed by Jarmo van Lenthe)

3.1.0 (2017-08-06)

New features:

- Support for exFAT filesystems (contributed by ldgriffin)
- Addition of `force_clean` method to `ImageParser`
- Addition of `--skip`, to complement `--only-mount`
- Improved support for UFS / BSD volumes

Bugfixes:

- Updated `--keys` to not parse commas (as these may be part of the key themselves) and properly support the asterisk key.
- Several fixes for `fsstat` call, including actually killing it after 3 seconds and optimizations by passing in the correct FS type, making mounting actually a lot faster.
- Fixes for path expanding (contributed by ruzzle)
- Re-index loopbacks prior to unmounting in the `Unmounter` class (contributed by ldgriffin)
- Use `--sizelimit` for various mounts, for support in newer kernel versions (contributed by ldgriffin)
- Improved support for non-ASCII volume labels

- The parted detection method does not hang anymore when parted requests input
- Fix for communication of LUKS keys to the `cryptsetup` command
- Fixes for reconstruction when multiple roots exist

3.0.1 (2017-04-08)

- Add support for qcow2 (contributed by Jarmo van Lenthe)
- Allow use of lowercase e01 file extension when mounting a directory in imount CLI (contributed by sourcex)
- Add ADS support for NTFS volumes (contributed by Patrick Leedom)
- Ability to lazily call fusermount -uz when unmounting (contributed by Patrick Leedom)
- Fix regression in mounting LV volumes; the path was incorrectly detected in `get_raw_path()` for these volumes.
- Fix regression in detection of single volumes that would be detected as DOS/MBR based on file type.

3.0.0 (2016-12-11)

This new release includes several backwards-incompatible changes, mostly because features were removed from the public API or have been renamed to obtain a more consistent API.

It was released after a long time of development, and does not even contain all features that were originally planned to go into this release, but it contains some important bugfixes that warranted a release.

New major features:

- Add volume shadow copy support for NTFS
- Add BDE (Bitlocker) support
- Addition of `--keys` CLI argument and corresponding argument to Volume class, allowing to specify key material for crypto mounts, supporting both BDE and LUKS.
- (Experimental) support for volume systems inside a volume. This is useful when e.g. a LVM volume contains in itself a MBR.
- A split between detection and initialization of volumes has been made. The basic way to access volumes as calling `init()`, but that mounted all volumes immediately. Now, `detect_*` methods have been added.
- Support `blkid` to retrieve FS type info
- Support for Linux RAID volumes
- (Still in development) interactive console, which will eventually become the primary means to interact with imagemounter.

Bugfixes:

- Calling `init()` will not automatically mount the volume when it is not `alloc`.
- Fix a bug where `.e01` files (lowercase) would not be recognized as Encase
- Fixed support for newer versions of `mmls`
- Fixed support for `pytsk3` under Python 3 (contributed by `insomniacslk`)
- Fixed support for EnCase v7 (EX01) image files (contributed by `pix`)
- Improved detection of several volume types

- `index` is now always `str`
- `Volume.size` is now always `int`
- Improved the unmounter with generic loopback support

Removed and modified features:

- Stopped providing `None` and `False` results when things go wrong for most methods. Instead, numerous exceptions have been added. These exceptions should be caught instead, or when using `mount_volumes` or `init`, you can specify `swallow_exceptions` (default) to restore previous behaviour. This is useful, since iteration will continue regardless of exceptions.
- Moved the attributes `fstypes`, `vstypes`, `keys`, `mountdir` and `pretty` to the `ImageParser` instance, so it does not need to get passed down through the `*args` hack anymore. For instance, `fstypes` has been moved; the dict will be inspected upon `Volume` instantiation and stored in the `fstype` attribute. Other arguments and attributes have been eliminated completely, or have been replaced by arguments to specific methods.
- Added an intermediary class `VolumeSystem`. Both `Volume` and `Disk` now use this (iterable) base class in their `volumes` attribute. If you relied on `volumes` being a list, you should now use `list(volumes)`. If you relied on indexing of the attribute, you could now also use `disk[0]` or `volume[0]` for finding the correct volume index. `volume_source` was moved to this class, as have `vstype` and `volume_detector`.
- **Changes to the CLI:**
 - Removed `--fsforce` and `--fsfallback`. Use `*` and `?` as `fstypes` instead for the same effect. This should make the CLI more sensible, especially regarding the `--fsforce` argument. The default FS fallback is still `unknown`, which can only be overridden by specifying `--fstypes=?=none`. (You can now specify `--fstypes=TYPE`, which equals to `--fstypes==TYPE`)
 - Removed `--stats` and `--no-stats`. These only complicated things and `fsstat` has been working fine for years now.
 - Removed `--raid` and `--no-raid` (due to Volume RAID support)
 - Removed `--disktype` and `--no-disktype`.
 - Renamed `--method` to `--disk-mounter`.
 - Renamed `--detection` to `--volume-detector`.
 - Renamed `--vstype` to `--vstypes`, now accepting a dict, similar to `--fstypes`
 - Moved the `imount.py` file into a new `cli` module, where also a new experimental shell-style CLI is under development.
- **Changes specific to `ImageParser`:**
 - Added `add_disk` and made `paths` optional in constructor.
 - Added indexing of the `ImageParser` and added `get_volume_by_index` method.
 - Removed `mount_single_volume` and `mount_multiple_volumes`. Use `init_volumes` instead, or use a custom loop for more control.
 - Dropped support for a single string argument for `paths` in `__init__`. Additionally, dropped the `paths` attribute entirely.
- **Changes specific to `Disk`:**
 - Renamed method to `disk_mounter` (see also CLI)
 - Removed `name`, `avfs_mountpoint` and `md_device` from public API.

- Removed Linux RAID Disk support. Instead, mount as a single volume, with the type of this volume being RAID. This greatly simplifies the `Disk` class. (This means that `loopback` has also been dropped from `Disk`)
- Added `detect_volumes` method, which can be used to detect volumes.
- Removed most `mount_*` methods. Moved `mount_volumes` to `init_volumes`. Functionality from the other methods can be restored with only a few lines of code.
- Removed the need for the rather obscure `multifile` attribute of `mount`. Only `xmount` actually required this, so we just implicitly use it there.
- Moved the `type` attribute to a method `get_disk_type`.
- **Changes specific to `Volume`:**
 - Renamed `get_raw_base_path` to `get_raw_path`
 - Renamed `get_size_gib` to `get_formatted_size`
 - Removed `get_magic_type`, `fill_stats`, `open_jffs2`, `find_lvm_volumes` and `open_luks_container` from public API.
 - Removed the `*_path`, `carvepoint` and `bindmountpoint` attributes from the public API. For `carvepoint`, the `carve` method now returns the path to the carvepoint. All data has been moved to the private `_paths` attribute. The `mountpoint` and `loopback` attributes are kept.
 - Removed `fsforce` and `fsfallback` arguments and attributes from `Volume` (see also CLI)
 - Added `init_volume`, which only mounts the single volume. It is used by `init` and the preferred way of mounting a single volume (instead of using `mount`)
 - Moved several attributes of `Volume` to a new `info` attribute, which is publicly accessible, but its contents are not part of a stable public API.
- **Changes specific to `VolumeSystem` (if you consider it on par with the functionality moved from `Disk`):**
 - Renamed `detection` to `volume_detector` (see also CLI)
 - Added a `VolumeSystem.detect_volumes()` iterable, which is the basic functionality of this class.
 - Moved `mount_single_volume` code from `Disk` to this class, adding the single volume detection method. The directory detection method has been incorporated in this new method.
- Dropped support for Python 3.2, since everyone seems to be doing that these days.

2.0.4 (2016-03-15)

- Add HFS+ support

2.0.3 (2015-08-02)

- Remove error prefix `([-])` from some of the warnings
- Do not warn about using `unknown` as `fsfallback` anymore
- Also work properly with the `python-magic` system package (in addition to the totally different `python-magic` PyPI package)
- *vmware-mount* Add `-r` to `vmware-mount` for readonly mounts

- *ntfs* Add force to mount options

2.0.2 (2015-06-17)

- Bugfix in *--check* regarding the *python-magic* module
- *vmware-mount* Fix vmware-mount support

2.0.1 (2015-06-17)

- Changed the default *fsfallback* to *unknown*, instead of *none*.

2.0.0 (2015-06-17)

- Introduce support for XFS, ISO, JFFS2, FAT, SquashFS, CramFS, VMFS, UDF and Minix (cheers martinvw!)
- Add ability to read the disk GUID using *disktype*, and read the filesystem magic for better detection of filesystems (cheers martinvw!)
- Add support for ‘mounting’ directories and compressed files using *avfs* (cheers martinvw!)
- Add support for detecting volumes using *parted*
- Introduce facility to carve filesystems for removed files, even in unallocated spaces
- Add *--no-interaction* for scripted access to the CLI
- Add *--check* for access to an overview of all dependencies of *imagemounter*
- Add *--casename* (and corresponding Python argument) to easily recognize and organize multiple mounts on the same system
- Change *--clean* to *--unmount*, supporting arguments such as *--mountdir* and *--pretty*, and made the code more robust and easier to read and extend
- Detect terminal color support and show color by default
- BSD is now called UFS
- *--stats* is now the default in the Python script
- NTFS mount now also shows the system files by default
- Do not stop when not running as root, but warn and probably fail miserably later on
- *fstype* now stores the detected file system type, instead of the *fstype* as determined by *fill_stats()*
- Logging now properly uses the Python logging framework, and there are now 4 verbosity levels
- Changes to how the pretty names are formatted
- Some Py2/Py3 compatibility fixes

1.5.3 (2015-04-08)

- Add support for *vmware-mount*

1.5.2 (2015-04-08)

- Ensure `Volume.size` is always int
- Fixed a GPT/DOS bug caused by TSK
- Add FAT support

1.5.1 (2014-05-22)

- Add disk index for multi-disk mounts

1.5.0 (2014-05-14)

- Add support for volume detection using mmls
- Python 3 support
- Bugfix in `luksOpen`

1.4.3 (2014-04-26)

- Experimental LUKS support

1.4.2 (2014-04-26)

- Bugfix that would prevent proper unmounting

1.4.1 (2014-02-10)

- Initial Py3K support
- Included script is now called `imount` instead of `mount_images`

1.4.0 (2014-02-03)

- `Disk` is now a separate class
- Some huge refactoring
- Numerous bugfixes, including resolving issues with unmounting
- Rename `image_mounter` to `imagemounter`
- Remove `mount_images` alias

1.3.1 (2014-01-23)

- More verbosity with respect to failing mounts

1.3.0 (2014-01-23)

- Add support for single volume mounts
- Add support for dummy base mounting
- Add support for RAID detection and mounting

1.2.9 (2014-01-21)

- Improve support for some types of disk images
- Some changes in the way some command-line arguments work (removed `-vs`, `-fs` and `-fsf`)

1.2.8 (2014-01-08)

- Make `:option:--stats` the default
- Print the volume size and offset in verbose mode in the CLI
- Add `imount` as command line utility name

1.2.7 (2014-01-08)

- Add `--keep`

1.2.6 (2014-01-08)

- Use fallback commands for base image mounting if the normal one fails
- Add `multifile` option to `Volume` to control whether `multifile` argument passing should be attempted
- Fix error in backwards compatibility of `mount_partitions`
- Copy the label of a volume to the last mountpoint if it looks like a mountpoint

1.2.5 (2014-01-07)

- Ability to automatically detect the mountpoint based on files in the filesystem

1.2.4 (2013-12-16)

- Partition is now `Volume`
- Store the volume flag (`alloc`, `unalloc`, `meta`)

1.2.3 (2013-12-10)

- Add support for pretty mount point names

1.2.2 (2013-12-09)

- Fix issue where 'extended' is detected as `ext` (again)

1.2.1 (2013-12-09)

- Fix issue where ‘extended’ is detected as ext
- ImagePartition is now Volume

1.2.0 (2013-12-05)

- ImagePartition is now responsible for mounting and obtaining its stats, and detecting lvm volumes
- LVM partitions are now mounted using this new mount method
- Utilize the partition size for disk size, which is more reliable
- Renamed ImagePartition to Volume (no backwards compatibility is provided)
- Add unknown mount type, for use with `--fstype`, which mounts without knowing anything
- Support mounting a directory containing *.001/*.E01 files

1.1.2 (2013-12-05)

- Resolve bug with respect to determining free loopback device

1.1.1 (2013-12-04)

- Improve `--clean` by showing the commands to be executed beforehand

1.1.0 (2013-12-04)

- Do not add sudo to internal commands anymore
- `--loopback` is removed, detects it automatically now
- `--clean` is added; will remove all traces of an unsuccessful previous run

1.0.4 (2013-12-03)

- Add the any vstype
- Fix some errors in the `mount_images` script

1.0.3 (2013-12-02)

- Support forcing the fstype
- Improved LVM support
- Added some warnings to CLI

1.0.2 (2013-11-28)

- Improved NTFS support

1.0.1 (2013-11-28)

- `command_exists` now works properly

1.0.0 (2013-11-28)

- Now includes proper `setup.py` and versioning
- Add support for reconstructing the filesystem using bindmounts
- More reliable use of `fsstat`
- Overhauled Python API with more transparency and less CLI requirements
 - Store yielded information in a `ImagePartition`
 - Remove dependency on `args` and add them to the class explicitly
 - Do not depend on user interaction or CLI output in `ImageParser` or `util`, but do CLI in `__main__`
- Support for LVM
- Support for `ewfmount`
- Retrieve stats more reliably
- New CLI arguments:
 - Colored output with `--color`
 - Wait for warnings with `--wait`
 - Support for automatic method with `--method=auto`
 - Specify custom mount dir with `--mountdir`
 - Specify explicit volume system type with `--vstype`
 - Specify explicit file system type with `--fstype`
 - Specify loopback device with `--loopback` (required by LVM support)

i

imagemounter, 9

Symbols

-carve
 command line option, 6
-casename
 command line option, 7
-check
 command line option, 5
-color
 command line option, 6
-disk-mounter <method>
 command line option, 7
-fstypes <types>
 command line option, 7
-help
 command line option, 5
-keep
 command line option, 6
-keys <keys>
 command line option, 7
-lazy-unmount
 command line option, 8
-mountdir <directory>
 command line option, 6
-no-color
 command line option, 6
-no-interaction
 command line option, 6
-no-single
 command line option, 8
-only-mount
 command line option, 6
-pretty
 command line option, 6
-read-write
 command line option, 7
-reconstruct
 command line option, 6
-single
 command line option, 8

-skip
 command line option, 6
-unmount
 command line option, 5
-verbose
 command line option, 6
-version
 command line option, 5
-volume-detector <method>
 command line option, 7
-vshadow
 command line option, 6
-vstypes <types>
 command line option, 7
-wait
 command line option, 6
-cn
 command line option, 7
-d <method>
 command line option, 7
-h
 command line option, 5
-k
 command line option, 6
-m <method>
 command line option, 7
-md <directory>
 command line option, 6
-p
 command line option, 6
-r
 command line option, 6
-rw
 command line option, 7
-u
 command line option, 5
-v
 command line option, 6
-w
 command line option, 6

<image> [<image> ...]
command line option, 5

C

command line option

- carve, 6
- casename, 7
- check, 5
- color, 6
- disk-mounter <method>, 7
- fstypes <types>, 7
- help, 5
- keep, 6
- keys <keys>, 7
- lazy-unmount, 8
- mountdir <directory>, 6
- no-color, 6
- no-interaction, 6
- no-single, 8
- only-mount, 6
- pretty, 6
- read-write, 7
- reconstruct, 6
- single, 8
- skip, 6
- unmount, 5
- verbose, 6
- version, 5
- volume-detector <method>, 7
- vshadow, 6
- vstypes <types>, 7
- wait, 6
- cn, 7
- d <method>, 7
- h, 5
- k, 6
- m <method>, 7
- md <directory>, 6
- p, 6
- r, 6
- rw, 7
- u, 5
- v, 6
- w, 6
- <image> [<image> ...], 5

I

imagemounter (*module*), 9