NAME

wimapply − Apply a WIM image

SYNOPSIS

wimapply *WIMFILE* [*IMAGE*] *TARGET* [*OPTION*...]

DESCRIPTION

wimapply, or equivalently **wimlib-imagex apply**, extracts ("applies") an image, or all images, from the Windows Imaging (WIM) archive *WIMFILE*.

IMAGE specifies the image in *WIMFILE* to extract. It may be the 1-based index of an image, the name of an image, or the keyword "all" to specify all images. It may be omitted if *WIMFILE* contains only one image. You can use **wiminfo**(1) to list the images contained in *WIMFILE*.

TARGET specifies where to extract the image(s) to. If *TARGET* is a directory, then the image(s) will be extracted to that directory as per **DIRECTORY EXTRACTION (UNIX)** or **DIRECTORY EXTRACTION (WINDOWS)**. If *TARGET* does not exist, then a directory will be created there first. Alternatively, if *TARGET* specifies a UNIX block device, then the image will be extracted to it as described in **NTFS VOLUME EXTRACTION (UNIX)**.

Note that **wimapply** is designed to extract, or "apply", full WIM images. If you instead want to extract only certain files or directories from a WIM image, use **wimextract**(1) instead.

If *IMAGE* is "all", then all images in *WIMFILE* will be extracted into subdirectories of *TARGET* named after the images, falling back to the image index when an image has no name or an unusual name. This is not yet supported in **NTFS VOLUME EXTRACTION (UNIX)** mode.

If *WIMFILE* is "-", then the WIM is read from standard input rather than from disk. See **PIPABLE WIMS** for more information.

DIRECTORY EXTRACTION (UNIX)

On UNIX-like systems, a WIM image may be extracted to a directory. This mode has the limitation that NTFS or Windows-specific metadata will not be extracted. Although some concepts such as hard links, symbolic links, last access timestamps, and last modification timestamps will be translated to their UNIX equivalents, other metadata will be lost (with warnings given). Notably, the following types of metadata will *not* be extracted in this mode:

- Windows file attribute flags
- Windows security descriptors (e.g. file owners and DACLs)
- File creation timestamps
- Reparse points other than symbolic links and junction points
- Named data streams
- Short filenames (also known as 8.3 names or DOS names).
- Object IDs

These same limitations apply to **wimextract**. As such, this mode is most useful in situations where NTFS or Windows-specific metadata is unimportant, e.g. when wanting to extract specific files, or when doing file archiving only on UNIX-like systems, possibly in combination with **--unix-data**. When Windows-specific metadata is important, then either the **NTFS VOLUME EXTRACTION (UNIX)** mode should be used, or the Windows version of wimlib should be used (see **DIRECTORY EXTRACTION (WINDOWS)**).

NTFS VOLUME EXTRACTION (UNIX)

On UNIX-like systems, *TARGET* may also be specified as a block device (e.g. /dev/sda3) containing an unmounted NTFS volume. In this mode, **wimapply** uses libntfs-3g to apply the specified WIM image to the root directory of the NTFS volume. The target volume should be empty, e.g. newly created by **mkntfs**(8). In this mode, NTFS-specific and Windows-specific data and metadata will be extracted, including the following:

- All data streams of all files except encrypted files, including the unnamed data stream as well as all named data streams.
- Reparse points, including symbolic links, junction points, and other reparse points.
- File and directory creation, access, and modification timestamps, using the native NTFS resolution of 100 nanoseconds.
- Windows security descriptors, including all components (owner, group, DACL, and SACL).
- Windows file attribute flags
- All names of all files, including names in the Win32 namespace, DOS namespace, Win32+DOS namespace, and POSIX namespace. This includes hard links.
- Object IDs.

However, encrypted files will not be extracted.

Restoring extended attributes (EAs) is also not yet supported in this mode.

Regardless, since almost all information from the WIM image is restored in this mode, it is possible (and fully supported) to restore an image of an actual Windows installation using **wimapply** on a UNIX-like system as an alternative to using **wimapply** or DISM on Windows. In the **EXAMPLES** section below, there is an example of applying an image from an "install.wim" file as may be found in the Windows installation media.

Note that to actually boot Windows (Vista or later) from an applied "install.wim" image, you also need to mark the partition as "bootable" and set up various boot files, such as \BOOTMGR and \BOOT\BCD. The latter task is most easily accomplished by running bcdboot.exe from a live Windows system such as Windows PE, but there are other options as well.

Finally, note that this mode uses libntfs-3g directly, without going through the **ntfs-3g**(8) driver. Hence, there is no special support for applying a WIM image to a directory on which an NTFS filesystem has been mounted using **ntfs-3g**(8); you have to unmount it first. There is also no support for applying a WIM image to some subdirectory of the NTFS volume; you can only apply to the root directory.

DIRECTORY EXTRACTION (WINDOWS)

On Windows, **wimapply** (and **wimextract**) natively support NTFS and Windows-specific metadata. For best results, the target directory should be located on an NTFS volume and the program should be run with Administrator privileges; however, non-NTFS filesystems and running without Administrator privileges are also supported, subject to limitations.

On Windows, **wimapply** tries to extract as much data and metadata as possible, including:

- All data streams of all files. This includes the default file contents, as well as named data streams if supported by the target volume.
- Reparse points, including symbolic links, junction points, and other reparse points, if supported by the target volume. Restoring symlinks requires Administrator privileges. Also see **--rpfix** and **--norpfix** for details on how absolute symbolic links and junctions are extracted.
- File and directory creation, access, and modification timestamps, to the highest resolution supported by the target volume.
- Security descriptors, if supported by the filesystem and **--no-acls** is not specified. Note that this, in general, requires Administrator privileges, and may be only partially successful if the program is run without Administrator privileges (see **--strict-acls**).
- File attribute flags, including hidden, compressed, encrypted, sparse, etc, when supported by the filesystem.
- Short filenames (also known as 8.3 names or DOS names).
- Hard links, if supported by the target filesystem.

- Object IDs, if supported by the target filesystem.
- Extended attributes (EAs), if supported by the target filesystem.

Additional notes about extracting files on Windows:

- **wimapply** will issue warnings if unable to extract the exact metadata and data of the WIM image due to limitations of the target filesystem.
- Since encrypted files (with FILE_ATTRIBUTE_ENCRYPTED) are not stored in plaintext in the WIM image, **wimapply** cannot restore encrypted files to filesystems not supporting encryption. Therefore, on such filesystems, encrypted files will not be extracted. Furthermore, even if encrypted files are restored to a filesystem that supports encryption, they will only be decryptable if the decryption key is available.
- Files with names that cannot be represented on Windows will not be extracted by default; see **--include-invalid-names**.
- Files with full paths over 260 characters (the so-called MAX_PATH) will be extracted, but beware that such files will be inaccessible to most Windows software and may not be able to be deleted easily.
- On Windows, unless the **--no-acls** option is specified, wimlib will attempt to restore files' security descriptors exactly as they are provided in the WIM image. Beware that typical Windows installations contain files whose security descriptors do not allow the Administrator to delete them. Therefore, such files will not be able to be deleted, or in some cases even read, after extracting, unless processed with a specialized program that knows to acquire the SE_RESTORE_NAME and/or SE_BACKUP_NAME privileges which allow overriding access control lists. This is not a bug in wimlib, which works as designed to correctly restore the data that was archived, but rather a problem with the access rights Windows uses on certain files. But if you just want the file data and don't care about security descriptors, use **--no-acls** to skip restoring all security descriptors.
- A similar caveat to the above applies to file attributes such as Readonly, Hidden, and System. By design, on Windows wimlib will restore such file attributes; therefore, extracted files may have those attributes. If this is not what you want, use the **--no-attributes** option.

SPLIT WIMS

You may use **wimapply** to apply images from a split WIM, or **wimextract** to extract files from a split WIM. The *WIMFILE* argument must specify the first part of the split WIM, while the additional parts of the split WIM must be specified in one or more **--ref**="*GLOB*" options. Since globbing is built into the **--ref** option, typically only one **--ref** option is necessary. For example, the names for the split WIM parts usually go something like:

```
mywim.swm
mywim2.swm
mywim3.swm
mywim4.swm
mywim5.swm
```

To apply the first image of this split WIM to the directory "dir", run:

```
wimapply mywim.swm 1 dir --ref="mywim*.swm"
```

PIPABLE WIMS

wimapply also supports applying a WIM from a nonseekable file, such as a pipe, provided that the WIM was captured in the wimlib-specific pipable format using **--pipable** (see **wimcapture**(1)). To use standard input as the WIM, specify "-" as *WIMFILE*. A possible use of this feature is to apply a WIM image being streamed from the network. For example, to apply the first image from a WIM file available on a HTTP server to an NTFS volume on /dev/sda1, run something like:

```
wget -O - http://myserver/mywim.wim | wimapply - 1 /dev/sda1
```

Pipable WIMs may also be split into multiple parts, just like normal WIMs. To apply a split pipable WIM from a pipe, the parts must be concatenated and all written to the pipe. The first part must be sent first, but

the remaining parts may be sent in any order.

OPTIONS

--check
Before applying the image, verify the integrity of *WIMFILE* if it has extra integrity information.

--ref="*GLOB*"
File glob of additional WIMs or split WIM parts to reference resources from. See **SPLIT_WIMS**. This option can be specified multiple times. Note: *GLOB* is listed in quotes because it is interpreted by **wimapply** and may need to be quoted to protect against shell expansion.

--rpfix, **--norpfix**
Set whether to fix targets of absolute symbolic links (reparse points in Windows terminology) or not. When enabled (**--rpfix**), extracted absolute symbolic links that are marked in the WIM image as being fixed are assumed to have absolute targets relative to the image root, and therefore **wimapply** prepends the absolute path to the extraction target directory to their targets. The intention is that you can apply an image containing absolute symbolic links and still have them be valid after it has been applied to any location.

The default behavior is **--rpfix** if any images in *WIMFILE* have been captured with reparse-point fixups done. Otherwise, it is **--norpfix**.

Reparse point fixups are never done in the NTFS volume extraction mode on UNIX-like systems.

--unix-data
(UNIX-like systems only) Restore UNIX-specific metadata and special files that were captured by **wimcapture** with the **--unix-data** option. This includes: standard UNIX file permissions (owner, group, and mode); device nodes, named pipes, and sockets; and extended attributes (Linux-only).

--no-acls
Do not restore security descriptors on extracted files and directories.

--strict-acls
Fail immediately if the full security descriptor of any file or directory cannot be set exactly as specified in the WIM file. If this option is not specified, when **wimapply** on Windows does not have permission to set a security descriptor on an extracted file, it falls back to setting it only partially (e.g. with SACL omitted), and in the worst case omits it entirely. However, this should only be a problem when running **wimapply** without Administrator rights. Also, on UNIX-like systems, this flag can also be combined with **--unix-data** to cause **wimapply** to issue an error if UNIX permissions are unable to be applied to an extracted file.

--no-attributes
Do not restore Windows file attributes such as readonly, hidden, etc.

--include-invalid-names
Extract files and directories with invalid names by replacing characters and appending a suffix rather than ignoring them. Exactly what is considered an "invalid" name is platform-dependent.

On POSIX-compliant systems, filenames are case-sensitive and may contain any byte except '\0' and ´/', so on a POSIX-compliant system this option will only have an effect in the unlikely case that the WIM image for some reason has a filename containing one of these characters.

On Windows, filenames are case-insensitive(*), cannot include control characters, and cannot include the characters '/', ´\0', '\', ':', '*', '?', ´"', '<', '>', or '|'. Ordinarily, files in WIM images should meet these conditions as well. However, it is not guaranteed, and in particular a WIM image captured with **wimcapture** on a POSIX-compliant system could contain such files. By default, invalid names will be ignored, and if there are multiple names differing only in case, one will be chosen to extract arbitrarily; however, with **--include-invalid-names**, all names will be sanitized and extracted in some form.

(*) Unless the ObCaseInsensitive setting has been set to 0 in the Windows registry, in which case certain software, including the Windows version of **wimapply**, will honor case-sensitive filenames

on NTFS and other compatible filesystems.

--wimboot
Windows only: Instead of extracting the files themselves, extract "pointer files" back to the WIM archive(s). This can result in significant space savings. However, it comes at several potential costs, such as not being able to delete the WIM archive(s) and possibly having slower access to files. See Microsoft's documentation for "WIMBoot" for more information.

If it exists, the [PrepopulateList] section of the file \Windows\System32\WimBootCompress.ini in the WIM image will be read. Files matching any of these patterns will be extracted normally, not as WIMBoot "pointer files". This is helpful for certain files that Windows needs to read early in the boot process.

This option only works when the program is run as an Administrator and the target volume is NTFS or another filesystem that supports reparse points.

In addition, this option works best when running on Windows 8.1 Update 1 or later, since that is the first version of Windows that contains the Windows Overlay Filesystem filter driver ("WOF"). If the WOF driver is detected, wimlib will create the WIMBoot "pointer files" using documented ioctls provided by WOF.

Otherwise, if the WOF driver is not detected, wimlib will create the reparse points and edit the file "\System Volume Information\WimOverlay.dat" on the target volume manually. This is potentially subject to problems, since although the code works in certain tested cases, neither of these data formats is actually documented by Microsoft. Before overwriting this file, wimlib will save the previous version in "\System Volume Information\WimOverlay.wimlib_backup", which you potentially could restore if you needed to.

You actually can still do a **--wimboot** extraction even if the WIM image is not marked as "WIMBoot-compatible". This option causes the extracted files to be set as "externally backed" by the WIM file. Microsoft's driver which implements this "external backing" functionality seemingly does not care whether the image(s) in the WIM are really marked as WIMBoot-compatible. Therefore, the "WIMBoot-compatible" tag (<WIMBOOT> in the XML data) seems to be a marker for intent only. In addition, the Microsoft driver can externally back files from WIM files that use XPRESS chunks of size 8192, 16384, and 32768, or LZX chunks of size 32768, in addition to the default XPRESS chunks of size 4096 that are created when **wimcapture** is run with the **--wimboot** option.

--compact=*FORMAT*
Windows-only: compress the extracted files using System Compression, when possible. This only works on either Windows 10 or later, or on an older Windows to which Microsoft's wofadk.sys driver has been added. Several different compression formats may be used with System Compression, and one must be specified as *FORMAT*. The choices are: xpress4k, xpress8k, xpress16k, and lzx.

Exclusions are handled in the same way as with the **--wimboot** option. That is: if it exists, the [PrepopulateList] section of the file \Windows\System32\WimBootCompress.ini in the WIM image will be read, and files matching any of the patterns in this section will not be compressed. In addition, wimlib has a hardcoded list of files for which it knows, for compatibility with the Windows bootloader, to override the requested compression format.

NOTES

Data integrity: WIM files include checksums of file data. To detect accidental (non-malicious) data corruption, wimlib calculates the checksum of every file it extracts and issues an error if it does not have the expected value. (This default behavior seems equivalent to the **/verify** option of ImageX.) In addition, a WIM file can include an integrity table (extra checksums) over the raw data of the entire WIM file. For performance reasons wimlib does not check the integrity table by default, but the **--check** option can be passed to make it do so.

ESD files: wimlib can extract files from solid-compressed WIMs, or "ESD" (.esd) files, just like from

normal WIM (.wim) files. However, Microsoft sometimes distributes ESD files with encrypted segments; wimlib cannot extract such files until they are first decrypted.

Security: wimlib has been carefully written to validate all input and is believed to be secure against some types of attacks which often plague other file archiving programs, e.g. directory traversal attacks (which, as it happens, Microsoft's WIM software is vulnerable to). Important parts of wimlib, e.g. the decompressors, have also been fuzz tested. However, wimlib is not currently designed to protect against some types of denial-of-service (DOS) attacks, e.g. memory exhaustion or "zip bombs".

EXAMPLES

Extract the first image from the Windows PE WIM on the Windows installation media to the directory "boot":

```
wimapply /mnt/windows/sources/boot.wim 1 boot
```

On Windows, apply an image of an entire volume, for example from "install.wim" which can be found on the Windows installation media:

```
wimapply install.wim 1 E:\
```

Same as above, but running on a UNIX-like system where the corresponding partition is /dev/sda2:

```
wimapply install.wim 1 /dev/sda2
```

Note that before running either of the above commands, an NTFS filesystem may need to be created on the partition, for example with format.exe on Windows or **mkntfs**(8) on UNIX-like systems. For example, on UNIX you might run:

```
mkntfs /dev/sda2 && wimapply install.wim 1 /dev/sda2
```

(Of course don't do that if you don't want to destroy all existing data on the partition!)

See **SPLIT WIMS** and **PIPABLE WIMS** for examples of applying split and pipable WIMs, respectively.

SEE ALSO

wimlib-imagex(1) **wimcapture**(1) **wimextract**(1) **wiminfo**(1)