## NAME

wimlib-imagex-apply − Extract one image, or all images, from a WIM archive

## SYNOPSIS

**wimlib-imagex apply** *WIMFILE* [*IMAGE*] *TARGET* [*OPTION*...]

## DESCRIPTION

**wimlib-imagex apply** extracts an image, or all images, from the Windows Imaging (WIM) file *WIMFILE*. This command is also available as simply **wimapply** if the appropriate hard link or batch file has been installed.

This command is designed to extract, or "apply", one or more full WIM images. If you instead want to extract only certain files or directories contained in a WIM image, consider using **wimlib-imagex extract** or **wimlib-imagex mount** instead. (**wimlib-imagex mount** is not supported on Windows.)

*IMAGE* specifies the WIM image in *WIMFILE* to extract. It may be a 1-based index of an image in *WIMFILE*, the name of an image in *WIMFILE*, or the keyword "all" to indicate that all images in *WIMFILE* are to be extracted. Use the **wimlib-imagex info** (1) command to show what images a WIM file contains. *IMAGE* may be omitted if *WIMFILE* contains only one image.

*TARGET* specifies where to extract the WIM image to. If *TARGET* specifies a directory, the WIM image is extracted to that directory (see **DIRECTORY EXTRACTION (UNIX)** or **DIRECTORY EXTRACTION (WINDOWS)**). Similarly, if *TARGET* specifies a non-existent file, a directory is created in that location and the WIM image is extracted to that directory.

If *IMAGE* is specified as "all", then all the images in *WIMFILE* are actually extracted into subdirectories of *TARGET*, each of which is given the name of the corresponding image, falling back to the image index in the case of an image with no name or a name not valid as a filename.

Alternatively, on UNIX-like systems only, if *TARGET* specifies a regular file or block device, it is interpreted as an NTFS volume to which the WIM image is to be extracted (see **NTFS VOLUME EXTRACTION (UNIX)**). Only a single image can be extracted in this mode, and only extracting to the root of the NTFS volume (not a subdirectory thereof) is supported.

*WIMFILE* may be "-" to read the WIM from standard input rather than from a file, but see **PIPABLE WIMS** for more information.

**wimlib-imagex apply** supports applying images from stand-alone WIMs as well as split WIMs. See **SPLIT WIMS**.

## DIRECTORY EXTRACTION (UNIX)

This section documents how **wimlib-imagex apply** (and also **wimlib-imagex extract**) extract a WIM image (or a possibly a subset thereof, in the case of **wimlib-imagex extract**) to a directory on UNIX-like systems. See **DIRECTORY EXTRACTION (WINDOWS)** for the corresponding documentation for Windows.

As mentioned, a WIM image can be applied to a directory on a UNIX-like system by providing a *TARGET* directory. However, it is important to keep in mind that the WIM format was designed for Windows, and as a result WIM files can contain data or metadata that cannot be represented on UNIX-like systems. The main information that **wimlib-imagex** will *not* be able to extract on UNIX-like systems is the following:

- Windows security descriptors (which include the file owner, group, and ACLs).
- Named data streams.
- Reparse points other than symbolic links and junction points.
- Certain file attributes such as compression, encryption, and sparseness.
- Short (DOS) names for files.
- File creation timestamps.

Notes: Unsupported data and metadata is simply not extracted, but **wimlib-imagex** will attempt to warn you when the contents of the WIM image can’t be exactly represented when extracted. Last access and last modification timestamps are specified to 100 nanosecond granularity in the WIM file, but will only be extracted to the highest precision supported by the underlying operating system, C library, and filesystem. Compressed files will be extracted as uncompressed, while encrypted files will not be extracted at all.

## NTFS VOLUME EXTRACTION (UNIX)

This section documents how **wimlib-imagex apply** extracts a WIM image directly to an NTFS volume image on UNIX-like systems.

As mentioned, **wimlib-imagex** running on a UNIX-like system can apply a WIM image directly to an NTFS volume by specifying *TARGET* as a regular file or block device containing an NTFS filesystem. The NTFS filesystem need not be empty, although it's expected that it be empty for the intended use cases. A new NTFS filesystem can be created using the **mkntfs**(8) command provided with **ntfs-3g**.

In this NTFS volume extraction mode, the WIM image is extracted to the root of the NTFS volume in a way preserves almost all information contained in the WIM image. It therefore does not suffer from the limitations described in **DIRECTORY EXTRACTION (UNIX)**. This support relies on libntfs-3g to write to the NTFS volume and handle NTFS-specific and Windows-specific data.

Please note that this NTFS volume extraction mode is *not* entered if *TARGET* is a directory, even if an NTFS filesystem is mounted on *TARGET*. You must specify the NTFS volume itself (and it must be unmounted, and you must have permission to write to it).

This NTFS volume extraction mode attempts to extract as much information as possible, including:

- 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).
- DOS/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.

However, there are also several known limitations of the NTFS volume extraction mode:

- Encrypted files will not be extracted.
- wimlib v1.7.0 and later: Sparse file attributes will not be extracted (same behavior as ImageX/DISM/WIMGAPI). wimlib v1.6.2 and earlier: Although sparse file attributes will be applied, the full data will be extracted to each sparse file, so extracted "sparse" files may not actually contain any sparse regions.

Regardless, since almost all information from the WIM image is restored in this mode, it is possible to restore an image of an actual Windows installation using **wimlib-imagex** on UNIX-like systems in addition to with **wimlib-imagex** on Windows. In the examples at the end of this manual page, there is an example of applying an image from the "install.wim" file contained in the installation media for Windows Vista, Windows 7, and Windows 8 in the "sources" directory.

But in order to actually boot Windows from an applied image, you must understand the boot process of Windows versions Vista and later. Basically, it is the following:

1. The Master Boot Record loads the Volume Boot Record (also called the Boot Sector) of the active partition, which is on an NTFS filesystem. This partition is called the "system partition".
2. The "bootmgr" program on the "system partition" is loaded (\BOOTMGR).
3. bootmgr loads the Boot Configuration Data (\Boot\BCD) from the "system partition".
4. Based on the information contained in the Boot Configuration Data, a loader for the Windows kernel is executed from the "Boot" partition, which is where Windows is installed.

So let's say you applied an image from an existing "install.wim" as in the example, or you've applied a custom Windows image that you've created using the **wimlib-imagex capture** (1) command. You've just applied the "Boot" partition, or the main Windows partition, but there is no "System" partition yet (i.e. no \BOOTMGR and no \Boot\BCD).

A "System" partition can be created created by running the "bcdboot.exe" program from within Windows or Windows PE. Alternatively, you can capture a separate WIM image containing the "System" partition. Or, the "System" partition may the same as the "Boot" partition, so the two "partitions" may be combined in one WIM image. However, as the \Boot\BCD file contains the Windows bootloader configuration, a WIM containing it can only be used on systems where you are setting up the same bootloader configuration, including the same partition layout.

Besides setting up the files on the "System" partition, don't forget to set the bootable flag on it, and have a master boot record that loads the bootable partition (Windows' MBR does, and SYSLINUX provides an equivalent MBR).

## DIRECTORY EXTRACTION (WINDOWS)

On Windows, **wimlib-imagex apply** and **wimlib-imagex extract** natively support Windows-specific and NTFS-specific data. For best results, the target directory should be located on an NTFS volume and **wimlib-imagex** should be run with Administrator privileges; however, non-NTFS filesystems and running without Administrator privileges are also supported.

On Windows, **wimlib-imagex apply** and **wimlib-imagex extract** try 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. (Note: see **--rpfix** and **--norpfix** for documentation on exactly how absolute symbolic links and junctions are extracted.) However, as per the default security settings of Windows, it is impossible to create a symbolic link or junction point without Administrator privileges; therefore, you must run **wimlib-imagex** as the Administrator if you wish to fully restore an image containing symbolic links and/or junction points. (Otherwise, merely a warning will be issued when a symbolic link or junction point cannot be extracted due to insufficient privileges.)
- 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. Furthermore, unless **--strict-acls** is specified, the security descriptors for individual files or directories may be omitted or only partially set if the user does not have permission to set them, which can be a problem if **wimlib-imagex** is run as a non-Administrator.
- File attributes, including hidden, sparse, compressed, encrypted, etc, when supported by the filesystem.
- DOS names (8.3) names of files; however, the failure to set them is not considered an error condition.
- Hard links, if supported by the filesystem.

Additional notes about extracting files on Windows:

- **wimlib-imagex** will issue a warning when it is unable to extract the exact metadata and data of the WIM image, for example due to features mentioned above not being supported by the target filesystem.
- Since encrypted files (with FILE_ATTRIBUTE_ENCRYPTED) are not stored in plaintext in the WIM image, **wimlib-imagex** 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 **wimlib-imagex apply** to apply images 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:

```
wimlib-imagex apply mywim.swm 1 dir --ref="mywim*.swm"
```

As a special case, if you are applying an image from standard input from a split WIM that is also pipable (as described in **PIPABLE WIMS**), the **--ref** option is unneeded; instead you must ensure that all the split WIM parts are concatenated together on standard input. They can be provided in any order, with the exception of the first part, which must be first.

## PIPABLE WIMS

As of wimlib 1.5.0, **wimlib-imagex apply** supports applying a WIM from a nonseekable file, such as a pipe, provided that the WIM was captured with **--pipable** (see **wimlib-imagex capture**(1)). To use standard input as the WIM, specify "-" as *WIMFILE*. A useful use of this ability is to apply an image from a WIM while streaming it from a server. 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
```

(The above also used the **wimapply** abbreviation for **wimlib-imagex apply**.) Note: WIM files are *not* pipable by default; you have to explicitly capture them with **--pipable**, and they are *not* compatible with Microsoft's software. See **wimlib-imagex capture**(1) for more information.

It is possible to apply an image from a pipable WIM split into multiple parts; see **SPLIT WIMS**.

## OPTIONS

**--check**
When reading *WIMFILE*, verify its integrity if the integrity table is present.

**--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 **wimlib-imagex** 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 **wimlib-imagex apply** 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 owners, groups, modes, and device IDs (major and minor numbers) that were captured by **wimlib-imagex capture** with the **--unix-data** option. As of wimlib v1.7.0, you can backup and restore not only the standard UNIX file permission information, but also character device nodes, block device nodes, named pipes (FIFOs), and UNIX domain sockets.

**--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 **wimlib-imagex** 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 **wimlib-imagex** without Administrator rights. Also, on UNIX-like systems, this flag can also be combined with **--unix-data** to cause **wimlib-imagex** to fail immediately if the UNIX owner, group, or mode on an extracted file cannot be set for any reason.

**--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 the characters ’/’, ´\0’, ’\’, ’:’, ’*’, ’?’, ’"’, ’<’, ’>’, or ’|’, and cannot end with a space or period. Ordinarily, files in WIM images should meet these conditions as well. However, it is not guaranteed, and in particular a WIM image captured with **wimlib-imagex** 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.

**--wimboot**
Windows only: Instead of extracting the files themselves, extract "pointer files" back to the WIM archive. This can result in significant space savings. However, it comes at several potential costs, such as not being able to delete the WIM archive 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 File System 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 **wimlib-imagex capture** is run with the **--wimboot** option.

## NOTES

*Data integrity*: WIM files include SHA1 message digests for file data. **wimlib-imagex apply** calculates the SHA1 message digest of every file it extracts and issues an error if it is not equal to the SHA1 message digest provided in the WIM. (This default behavior seems equivalent to the **/verify** option of ImageX.) Note that this is separate from the integrity table of the WIM, which provides SHA1 message digests over raw chunks of the entire WIM file and is checked separately if the **--check** option is specified.

*ESD files*: wimlib v1.6.0 and later can extract files from version 3584 WIMs, which usually contain LZMS-compressed solid blocks and may carry the *.esd* file extension rather than *.wim*. However, *.esd* files downloaded directly by the Windows 8 web downloader have encrypted segments, and wimlib cannot extract such files until they are first decrypted.

*Directory traversal attacks*: wimlib validates filenames before extracting them and is not vulnerable to directory traversal attacks. This is in contrast to Microsoft WIMGAPI/ImageX/DISM which can overwrite arbitrary files on the target drive when extracting a malicious WIM file containing files named .. or containing path separators.

## EXAMPLES

Extract the first image from the Windows PE image on the Windows Vista/7/8 installation media to the directory "boot":

```
wimlib-imagex apply /mnt/windows/sources/boot.wim 1 boot
```

Same as above, but using the **wimapply** abbreviation:

```
wimapply /media/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 Vista/7/8 installation media:

```
wimlib-imagex apply install.wim 1 E:\
```

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

```
wimlib-imagex apply 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) (part of NTFS-3g) on UNIX-like systems.

For example, 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!)

An example of applying a pipable WIM from a pipe can be found in **PIPABLE WIMS**, and an example of applying a split WIM can be found in **SPLIT WIMS**.

## SEE ALSO

**wimlib-imagex**(1) **wimlib-imagex-capture**(1) **wimlib-imagex-extract**(1) **wimlib-imagex-info**(1)