NAME

wimlib-imagex-capture, wimlib-imagex-append − Create or append a WIM image

SYNOPSIS

wimlib-imagex capture *SOURCE WIMFILE* [*IMAGE_NAME* [*IMAGE_DESCRIPTION*]] [*OPTION*...]
wimlib-imagex append *SOURCE WIMFILE* [*IMAGE_NAME* [*IMAGE_DESCRIPTION*]] [*OPTION*...]

DESCRIPTION

The **wimlib-imagex capture** and **wimlib-imagex append** commands create a Windows Imaging (WIM) image from a directory tree. The **wimlib-imagex capture** command creates a new WIM file containing the captured image, while the **wimlib-imagex append** command appends the captured image to an existing WIM file. These commands are also available as simply **wimcapture** and **wimappend** if the appropriate hard links or batch files are installed.

Background information: A WIM image is an independent directory tree in a WIM file. A WIM file may contain any number of separate images. WIM files are single-instancing with regards to file data, so a file is stored only one time in the entire WIM, regardless of how many images the file appears in.

SOURCE specifies the location of the files to create the new WIM image from. If *SOURCE* is a directory, the WIM image is captured from that directory (see **DIRECTORY CAPTURE (UNIX)** or **DIRECTORY CAPTURE (WINDOWS)**). Alternatively, if the **--source-list** option is specified, *SOURCE* is interpreted as a file that itself provides a list of files and directories to include in the new WIM image. Still alternatively, only on UNIX-like systems, if *SOURCE* is a regular file or block device, it is interpreted as an NTFS volume from which a WIM image is to be captured using libntfs-3g (see **NTFS VOLUME CAPTURE (UNIX)**).

IMAGE_NAME and *IMAGE_DESCRIPTION* specify the name and description to give the new WIM image. If *IMAGE_NAME* is not specified, it defaults to the base name (excluding path to parent directory) of *SOURCE*, but if this name already exists in *WIMFILE*, a unique suffix is added. Otherwise, *IMAGE_NAME* must be either a name that does not already exist as an image in *WIMFILE*, or the empty string to create an image with no name. If *IMAGE_DESCRIPTION* is not specified, no description is given to the new image.

As a special case, if *WIMFILE* is "-", the **--pipable** option is assumed and the WIM file is written to standard output in a special pipable format. See the documentation for **--pipable** for more details.

DIRECTORY CAPTURE (UNIX)

This section documents how **wimlib-imagex** captures files from a directory tree on UNIX-like systems. See **DIRECTORY CAPTURE (WINDOWS)** for the corresponding documentation for Windows.

On UNIX-like systems, when *SOURCE* specifies a directory or a symbolic link to a directory, the WIM image will be captured from the directory tree rooted at this directory. This directory can be on any type of filesystem, and mountpoints are followed recursively. In this mode, wimlib will store the following types of information:

- Directories and regular files, and the contents of regular files
- Hard links
- Symbolic links (translated losslessly to Windows reparse points)
- Last modification times (mtime) and last access times (atime) with 100 nanosecond granularity
- With **--unix-data**: UNIX owners, groups, and modes
- With **--unix-data**: device nodes, FIFOs, and UNIX domain sockets

There is no support for storing extended attributes (e.g. SELinux security labels and POSIX ACLs). Also note that last status change times (ctime) are not stored.

Pedantic note: A limitation of the WIM format prevents the unusual case where a single symbolic link file itself has multiple names (hard links); in this unlikely case, each symbolic link is stored as an independent file.

NTFS VOLUME CAPTURE (UNIX)

This section documents how **wimlib-imagex** captures files directly from an NTFS volume image on UNIX-like systems.

On UNIX-like systems, a special image capture mode is entered when *SOURCE* is a regular file or block device. In this mode, *SOURCE* is assumed to be an NTFS volume or volume image, and **wimlib-imagex** will capture a WIM image containing the full contents of the NTFS volume, including NTFS-specific data. This is done using libntfs-3g.

Note that the NTFS volume capture mode is *not* entered if *SOURCE* is a directory, even if an NTFS filesystem is mounted on *SOURCE* using ntfs-3g. You must specify the NTFS volume itself (and it must be unmounted, and you must have permission to read from it).

The NTFS volume capture mode attempts to capture as much data and metadata as possible, including:

- All data streams of all unencrypted 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, the main limitations of this NTFS volume capture mode are:

- Encrypted files are excluded by default. Although libntfs-3g can read their data, they need to be stored in the WIM file in a special format that wimlib does not yet support (except on Windows, where wimlib can treat the data as opaque and hand it off to the appropriate API function).
- The sparse attribute on sparse files will be saved, but the data stored will be the full data of the file rather than the "sparse" data. (The data is, however, subject to the WIM format's compression.)

DIRECTORY CAPTURE (WINDOWS)

On Windows, **wimlib-imagex capture** and **wimlib-imagex append** natively support Windows-specific and NTFS-specific data. They therefore act similarly to the corresponding commands of Microsoft's ImageX or DISM. For best results, the directory being captured should be 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 capture** and **wimlib-imagex append** try to archive as much data and metadata as possible, including:

- All data streams of all files.
- Reparse points, including symbolic links, junction points, and other reparse points, if supported by the source filesystem. (Note: see **--rpfix** and **--norpfix** for documentation on exactly how absolute symbolic links and junctions are captured.)
- File and directory creation, access, and modification timestamps. These are stored with Windows NT's native timestamp resolution of 100 nanoseconds.
- Security descriptors, if supported by the source filesystem and **--no-acls** is not specified. However, beware that unless **--strict-acls** is specified, the security descriptors for individual files or directories may be omitted or only partially captured if the user does not have permission to read them, which can be a problem if **wimlib-imagex** is run as a non-Administrator.
- File attributes, including hidden, sparse, compressed, encrypted, etc. Encrypted files will be stored in encrypted form rather than in plain text. Transparently compressed files will be read as uncompressed and stored subject to the WIM's own compression. There is no special handling for storing sparse

files, but they are likely to compress to a small size.

- DOS names (8.3) names of files; however, the failure to read them is not considered an error condition.
- Hard links, if supported by the source filesystem.

There is no support for storing NTFS extended attributes and object IDs.

The capture process is reversible, since when **wimlib-imagex apply** (on Windows) extracts the captured WIM image, it will extract all of the above information, at least to the extent supported by the destination filesystem.

Pedantic note: since Windows is not fully compatible with its own filesystem (NTFS), on Windows wimlib cannot archive certain files that may exist on a valid NTFS filesystem but are inaccessible to the Windows API, for example two files with names differing only in case in the same directory, or a file whose name contains certain characters considered invalid by Windows. If you run into problems archiving such files consider using the **NTFS VOLUME CAPTURE (UNIX)** mode from Linux.

OPTIONS

--boot

Specifies that the new image is to be made the bootable image of the WIM archive.

--check

For **wimlib-imagex append**, before performing the append operation, check the integrity of *WIMFILE* if an integrity table is present. Furthermore, include an integrity table in the new WIM file (**wimlib-imagex capture**) or the modified WIM file (**wimlib-imagex append**). If this option is not specified, no integrity table is included in a WIM file created with **wimlib-imagex capture**, while a WIM file updated with **wimlib-imagex append** will be written with an integrity table if and only if one was present before.

--compress=*TYPE*[:*LEVEL*]

Specifies the compression format for the new WIM file. *TYPE* may be "none", "XPRESS" (alias: "fast"), "LZX" (alias: "maximum"), or "LZMS" (alias: "recovery"). *TYPE* is matched case-insensitively. The default is "LZX".

You can optionally also specify an integer compression *LEVEL*. The compression level specifies how hard the compression algorithm for the specified compression *TYPE* will work to compress the data. The values are scaled so that 20 is quick compression, 50 is medium compression, and 100 is high compression. However, you can choose any value, and not just these particular values. The default is 50.

This option only affects the compression type used in non-solid WIM resources. If you are creating a solid WIM (using the **--solid** option), then you probably want **--solid-compress** instead.

Be careful if you choose LZMS compression. It is not compatible with wimlib before v1.6.0, WIMGAPI before Windows 8, DISM before Windows 8.1, and 7-Zip.

Also note that choosing LZMS compression does not automatically imply solid-mode compression, as it does with DISM. Use **--solid** if you want to create a solid WIM, or "ESD file".

--chunk-size=*SIZE*

Set the compression chunk size to *SIZE* bytes. A larger compression chunk size results in a better compression ratio. wimlib supports different chunk sizes depending on the compression type:

- XPRESS: 4K, 8K, 16K, 32K, 64K
- LZX: 32K, 64K, 128K, 256K, 512K, 1M, 2M
- LZMS: 32K, 64K, 128K, 256K, 512K, 1M, 2M, 4M, 8M, 16M, 32M, 64M, 128M, 256M, 512M, 1G

You can provide the full number (e.g. 32768), or you can use one of the K, M, or G suffixes. KiB, MiB, and GiB are also accepted.

This option only affects the chunk size used in non-solid WIM resources. If you are creating a solid WIM (using the **--solid** option), then you probably want **--solid-chunk-size** instead.

Use this option with caution if compatibility with Microsoft's implementation is desired, since their implementation has limited support for non-default chunk sizes.

--solid

Create a "solid" WIM file that compresses files together rather than independently. This results in a significantly better compression ratio, but it comes at the cost of various tradeoffs, including: slow compression with very high memory usage; slow random access to the resulting WIM file; and reduced compatibility.

Compatibility-wise, the first version of Microsoft's WIMGAPI to support solid WIM files was released with Windows 8, and the first version of DISM to do so was released with Windows 8.1.

If you want to create an "ESD file", then use this option. An (unencrypted) "ESD file" is a solid WIM file.

By default, this option has an effect equivalent to DISM's option **/compress:recovery**. The options for wimlib-imagex are different because they try not to conflate the compression type (e.g. LZX or LZMS) with solid-mode compression, as these are two different things.

--solid-chunk-size=*SIZE*

Like **--chunk-size**, but set the chunk size used in solid resources. The default, assuming LZMS compression, is 64MiB (67108864); this requires about 640MiB of memory per thread. This option only has an effect when **--solid** is also specified. Note: Microsoft's implementation is not compatible with LZMS chunk sizes larger than 64MiB.

--solid-compress=*TYPE*[:*LEVEL*]

Like **--compress**, but set the compression type used in solid resources. The default is LZMS compression. This option only has an effect when **--solid** is also specified.

--threads=*NUM_THREADS*

Number of threads to use for compressing data. Default: autodetect (number of available CPUs).

--rebuild

For **wimlib-imagex append**: rebuild the entire WIM rather than appending the new data to the end of it. Rebuilding the WIM is slower, but will save a little bit of space that would otherwise be left as a hole in the WIM. Also see **wimlib-imagex optimize**(1).

--flags=*EDITIONID*

Specify a string to use in the <FLAGS> element of the XML data for the new image.

--dereference

(UNIX-like systems only) Follow symbolic links and archive the files they point to, rather than archiving the links themselves.

--config=*FILE*

Specifies a configuration file (UTF-8 or UTF-16LE encoded; plain ASCII also works) for capturing the new image. The configuration file specifies files that are to be treated specially during the image capture.

The format of the configuration file is INI-style; that is, it is arranged in bracketed sections. Currently, the following sections are recognized:

- [ExclusionList] --- contains a list of path globs to exclude from capture. If a directory is matched, both the directory and its contents are excluded.
- [ExclusionException] --- contains a list of path globs to include in the capture, even when the file or directory also matches a glob in [ExclusionList].
- [PrepopulateList] --- this does not affect capture, but if the image is applied later with **--wimboot**, these are globs of files that shall be extracted normally, not as WIMBoot "pointer files". If a directory is matched, all files and subdirectories are also matched recursively.

Path globs may contain the '*' and '?' meta-characters. Relative globs (e.g. *.mp3) match against a filename in any directory. Absolute globs (e.g. /dir/file), are treated as paths starting at the main directory being captured, or the root of the NTFS volume for NTFS volume capture mode. Do not use drive letters in the paths; they will be ignored. Path separators may be either forwards slashes or backwards slashes.

Lines beginning with the '#' or ';' characters are treated as comments and ignored. Globs with whitespace in them need not be quoted; however, if they are, both double and single quotes are accepted.

If this option is not specified the following default configuration file is used:

```
[ExclusionList]
\$ntfs.log
\hiberfil.sys
\pagefile.sys
\swapfile.sys
\System Volume Information
\RECYCLER
\Windows\CSC
```

However, special behavior applies if **--wimboot** is also specified. By default, with **--wimboot** specified, the file Windows/System32/WimBootCompress.ini in the directory being captured will be used as the configuration file. However, this can be overridden using **--config**; and this also causes the specified configuration file to be saved in the WIM image as Windows/System32/WimBootCompress.ini, overriding any that may be present on the filesystem.

--unix-data
(UNIX-like systems only) Store the UNIX owner, group, mode, and device ID (major and minor number) of each captured file. 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.

wimlib stores UNIX data by adding a special tagged metadata item to each directory entry of each file that contains this information. This extra information is ignored by the Microsoft implementation. Note: UNIX data stored by wimlib before v1.7.0 used a different format that is no longer supported. If you have old WIM files with UNIX data, apply them with v1.6.2 and recapture them with v1.7.0 or later.

--no-acls
Do not capture files' security descriptors.

--strict-acls
Fail immediately if the full security descriptor of any file cannot be read. On Windows, the default behavior without this option is to first try omitting the SACL from the security descriptor, then to try omitting the security descriptor entirely. The purpose of this is to capture as much data as possible without always requiring Administrator privileges. However, if you desire that all security descriptors be captured exactly, you may wish to provide this option, although the Administrator should have permission to read everything anyway.

--rpfix, **--norpfix**
Set whether to fix targets of absolute symbolic links (reparse points in Windows terminology) or not. When enabled (**--rpfix**), absolute symbolic links that point inside the directory tree being captured will be adjusted to be absolute relative to the root of the directory tree being captured. When disabled (**--norpfix**), absolute symbolic links will be captured exactly as is.

The default behavior for **wimlib-imagex capture** is equivalent to **--rpfix**. The default behavior for **wimlib-imagex append** will be **--rpfix** if reparse point fixups have previously been done on *WIMFILE*, otherwise **--norpfix**.

In the case of a multi-source capture, (**--source-list** specified), passing **--norpfix** is recommended. Otherwise, reparse point fixups will be disabled on all capture sources destined for non-root locations in the WIM image, while capture sources destined for the WIM root will get the default behavior from the previous paragraph.

--source-list

wimlib-imagex capture and **wimlib-imagex append** support creating a WIM image from multiple separate files or directories. When **--source-list** is specified, the *SOURCE* argument specifies the name of a text file, each line of which is either 1 or 2 whitespace separated file paths. The first file path, the source, specifies the path to a file or directory to capture into the WIM image. It may be either absolute or relative to the current working directory. The second file path, if provided, is the target and specifies the path in the WIM image that this file or directory will be saved as. Leading and trailing slashes in the target are ignored, except if it consists entirely of slashes (e.g. "/"), which indicates that the directory is to become the root of the WIM image. If omitted, the target string defaults to the same as the source string.

An example source list file is as follows:

```
# Make the WIM image from the 'winpe' directory
winpe    /

# Send the 'overlay' directory to '/overlay' in the WIM image
overlay  /overlay

# Overlay a separate directory directly on the root of the WIM image.
/data/stuff          /
```

Subdirectories in the WIM are created as needed. Multiple source directories may share the same target, which implies an overlay. In the event that this results a nondirectory file being added to the WIM image multiple times, the last version (as listed in the source list file) overrides any earlier version.

File paths containing whitespace may be quoted with either single quotes or double quotes. Quotes may not be escaped.

Lines consisting only of whitespace and lines beginning with '#' preceded by optional whitespace are ignored.

As a special case, if *SOURCE* is "-", the source list is read from standard input rather than an external file.

The NTFS volume capture mode on UNIX-like systems cannot be used with **--source-list**, as only capturing a full NTFS volume is supported.

--pipable

Create a "pipable" WIM, which can be applied fully sequentially, including from a pipe. An image in the resulting WIM can be applied with **wimlib-imagex apply**, either normally by specifying the WIM file name, or with **wimlib-imagex apply -** to read the WIM from standard input. See **wimlib-imagex apply**(1) for more details.

For append operations, this option will result in a full rebuild of the WIM to make it pipable. For capture operations, the captured WIM is simply created as pipable. Beware that the more images you add to a pipable WIM, the less efficient piping it will be, since more unneeded data will be sent through the pipe.

When wimlib creates a pipable WIM, it carefully re-arranges the components of the WIM so that they can be read sequentially and also makes several other modifications. As a result, these "pipable" WIMs are *not compatible with Microsoft's software*, so keep this in mind if you're going to use them. If desired, you can use **wimlib-imagex optimize --not-pipable** to re-write a pipable WIM as a regular WIM. (**wimlib-imagex export** also provides the capability to export images from

a pipable WIM into a non-pipable WIM, or vice versa.)

For the most part, wimlib operates on pipable WIMs transparently. You can modify them, add or delete images, export images, and even create split pipable WIMs. The main disadvantages are that appending is (currently) less efficient (**--rebuild** is always implied), and also they aren't compatible with Microsoft's software.

wimlib-imagex capture and **wimlib-imagex append** can both write a pipable WIM directly to standard output; this is done automatically if *WIMFILE* is specified as "-". (In that case, **--pipable** is assumed.)

--not-pipable
Ensure the resulting WIM is in the normal, non-pipable WIM format. This is the default for **wimlib-imagex capture**, except when writing to standard output (*WIMFILE* specified as "-"), and also for **wimlib-imagex append**, except when appending to a WIM that is already pipable.

--update-of=[*WIMFILE*:]*IMAGE*
Declares that the image being captured or appended from *SOURCE* is mostly the same as the existing image *IMAGE* in *WIMFILE*, but captured at a later point in time, possibly with some modifications in the intervening time. This is designed to be used in incremental backups of the same filesystem or directory tree. *IMAGE* can be a 1-based index or name of an existing image in *WIMFILE*. It can also be a negative integer to index backwards into the images (e.g. -1 means the last existing image in *WIMFILE*).

When this option is provided, the capture or append of the new image will be optimized by not reading files that, based on metadata such as timestamps, appear not to have been modified since they were archived in the existing *IMAGE*. Barring manipulation of timestamps, this option only affects performance and does not change the resulting WIM image.

As shown, the full syntax for the argument to this option is to specify the WIM file, a colon, and the image; for example, "--update-of mywim.wim:1". However, the WIM file and colon may be omitted, in which case the WIM file will default to the WIM file being appended to for append operations, or the WIM file from which a delta is being taken (only if **--delta-from** is specified exactly once) for capture operations.

--delta-from=*WIMFILE*
For **wimlib-imagex capture** only: capture the new WIM as a "delta" from *WIMFILE*. Any streams that would ordinarily need to be archived in the new WIM are omitted if they are already present in the *WIMFILE* on which the delta is being based. The new WIM will still contain a full copy of the image metadata, but this is typically only a small fraction of a WIM's total size.

This option can be specified multiple times, in which case the resulting delta WIM will only contain streams not present in any of the specified base WIMs.

To operate on the resulting delta WIM using other commands such as **wimlib-imagex apply**, you must specify the delta WIM as the WIM file to operate on, but also reference the base WIM(s) using the **--ref** option. Beware: to retain the proper functioning of the delta WIM, you can only add, not delete, files and images to the base WIM(s) following the capture of a delta from it.

--delta-from may be combined with **--update-of** to increase the speed of capturing a delta WIM.

As an example, consider the following backup and restore sequence:

(initial backup)

```
$ wimcapture /some/directory bkup-base.wim
```

(some days later, create second backup as delta from first)

```
$ wimcapture /some/directory bkup-2013-08-20.dwm \
        --update-of bkup-base.wim:-1 --delta-from bkup-base.wim
```

(restoring the second backup)

```
$ wimapply bkup-2013-08-20.dwm --ref=bkup-base.wim 1 \
      /some/directory
```

However, note that as an alternative to the above sequence that used a delta WIM, the second backup could have simply been appended to the WIM as new image using **wimlib-imagex append**. Delta WIMs should be used only if it's desired to base the backups or images on a separate, large file that is rarely modified.

Note: unlike "pipable" WIMs (created with the **--pipable** option), "delta" WIMs (created with the **--delta-from** option) are compatible with Microsoft's software. For example, you can use the /ref option of ImageX to reference the base WIM(s), similar to above.

Additional note: **wimlib-imagex** is generalized enough that you can in fact combine **--pipable** and **--delta-from** to create pipable delta WIMs. In such cases, the base WIM(s) must be captured as pipable as well as the delta WIM, and when applying an image, the base WIM(s) must be sent over the pipe after the delta WIM.

--wimboot
Mark the image as WIMBoot-compatible. See Microsoft's documentation for more information about WIMBoot. This option will, by default, set the compression type to XPRESS and the chunk size to 4096 bytes; these can, however, still be overridden through the **--compress** and **--chunk-size** parameters, respectively. In addition, this option will, by default, set the configuration file to *SOURCE*\Windows\System32\WimBootCompress.ini if present and accessible; however, this may still be overridden through the **--config** parameter.

--unsafe-compact
See the documentation for this option in **wimlib-imagex-optimize** (1).

NOTES

wimlib-imagex append does not support appending an image to a split WIM.

Except when using **--unsafe-compact**, it is safe to abort a **wimlib-imagex append** command partway through; however, after doing this, it is recommended to run **wimlib-imagex optimize** to remove any data that was appended to the physical WIM file but not yet incorporated into the structure of the WIM, unless the WIM was being fully rebuilt (e.g. with **--rebuild**), in which case you should delete the temporary file left over.

wimlib-imagex creates WIMs compatible with Microsoft's software (WIMGAPI, ImageX, DISM), with some caveats:

- With **wimlib-imagex** on UNIX-like systems, it is possible to create a WIM image containing files with names differing only in case, or files with names containing the characters ':', '*', '?', '"', '<', '>', '|', or '\', which are valid on POSIX-compliant filesystems but not Windows. Be warned that such files will not be extracted by default by the Windows version of **wimlib-imagex**, and (even worse) Microsoft's ImageX can be confused by such names and quit extracting the image partway through. (It perhaps is worth pointing out that Windows' own default filesystem, NTFS, supports these characters, although Windows does not!)
- Pipable WIMs are incompatible with Microsoft's software. Pipable WIMs are created only if *WIMFILE* was specified as "-" (standard output) or if the **--pipable** flag was specified.
- WIMs captured with a non-default chunk size (with the **--chunk-size** option) or as solid archives (with the **--solid** option) or with LZMS compression (with **--compress**=LZMS or **--compress**=recovery) have varying levels of compatibility with Microsoft's software. Generally, more recent versions of Microsoft's software are more compatible.

EXAMPLES

First example: Create a new WIM 'mywim.wim' with LZX ("maximum") compression that will contain a captured image of the directory tree 'somedir'. Note that the image name need not be specified and will default to 'somedir':

wimlib-imagex capture somedir mywim.wim

or, if the **wimcapture** hard link or batch file has been installed, the abbreviated form can be used:

wimcapture somedir mywim.wim

The remaining examples will use the long form, however. Next, append the image of a different directory tree to the WIM created above:

wimlib-imagex append anotherdir mywim.wim

Easy enough, and the above examples of imaging directory trees work on both UNIX-like systems and Windows. Next, capture a WIM with several non-default options, including XPRESS ("fast") compression, an integrity table, no messing with absolute symbolic links, and an image name and description:

wimlib-imagex capture somedir mywim.wim --compress=fast \
--check --norpfix "Some Name" "Some Description"

Capture an entire NTFS volume into a new WIM file and name the image "Windows 7". On UNIX-like systems, this requires using the special mode described in **NTFS VOLUME CAPTURE (UNIX)** where *SOURCE* is a file or block device containing an NTFS filesystem:

wimlib-imagex capture /dev/sda2 windows7.wim "Windows 7"

or, on Windows, to capture a full NTFS volume you instead need to specify the root directory of the mounted volume, for example:

wimlib-imagex capture E:\ windows7.wim "Windows 7"

Same as above example with capturing an NTFS volume from **wimlib-imagex** running on a UNIX-like system, but capture the WIM in the wimlib-specific "pipable" format that can be piped to **wimlib-imagex apply**:

wimlib-imagex capture /dev/sda2 windows7.wim "Windows 7" \
--pipable

Same as above, but instead of writing the pipable WIM to the file "windows7.wim", write it directly to standard output through a pipe into some other program "someprog", which could, for example, be a program or script that streams the data to a server. Note that **--pipable** need not be explicitly specified when using standard output as the WIM "file":

wimlib-imagex capture /dev/sda2 - "Windows 7" | someprog

SEE ALSO

wimlib-imagex(1), **wimlib-imagex-apply**(1)