Using WavPack Version 5.5.0

WavPack consists of two complementary command-line programs, WavPack and WvUnpack. These programs allow you to compress (and restore) audio files in several formats including the Microsoft .wav format. The audio files must be uncompressed PCM (not ADPCM, for example), but other than that there are few restrictions. The files may be any resolution from 8 to 32 bits; they may be mono, stereo, or multichannel; they may even be IEEE floating point data or 1-bit DSD audio (in those formats that support DSD).

Two additional utilities are provided to manipulate metadata stored in APEv2 format tags located at the end of WavPack files. The first, WvGain, is used to apply (after loudness analysis) ReplayGain information to WavPack files and the other, WvTag, is a more general utility for appending or removing all kinds of other metadata items from WavPack files (including cover art and cuesheets).

The programs are initiated from the command line with the following syntax:

Usage:

```
WAVPACK [-options] infile[.wav]|infile.ext|- [outfile[.wv]|outpath|-]
WAVPACK --drop [-options] infile[.wav]|infile.ext [...]
WVUNPACK [-options] infile[.wv]|- [outfile[.ext]|outpath|-]
WVUNPACK --drop [-options] infile[.wv] [...]
WVGAIN [-options] infile[.wv] [...]
WVTAG [-options] infile[.wv] [...]
```

The filename extensions will be defaulted if not provided and the input filename may contain wildcards for doing batch operations. Specifying the output filename is optional, and if more than one input file is specified (with wildcards) then the only acceptable output specification is an output path. If the input filename begins with a '@' then the file is assumed to contain a list of newline separated filenames to be processed in batch mode (the output file may also be done this way although it may contain only a single line). The Windows frontend uses this mechanism for passing filenames, although it could be used for other purposes. Either filename can be replaced with "-" to allow use of stdin or stdout, although because the hybrid lossless mode uses two files it is not compatible with pipes.

The input files for WavPack may be either Microsoft .wav files (as mentioned above), or any of the other formats listed below, including existing WavPack files. They may also be raw PCM files (specified with the --raw-pcm option). In the case of reëncoding from existing WavPack files, all tags are copied (and may be modified with the tag specification options) and unless an alternate output name or directory is specified, the source files will be overwritten (safely). Used with the filename wildcards, this can be used to easily reëncode an entire directory of WavPack files with a single command. Reëncoding from lossy to lossless is not allowed, but other combinations like pure lossless to hybrid lossless (or lossy) are fine. Note that when reëncoding from a configuration with a correction file to a configuration without a correction file, the source correction file will *not* be automatically deleted even if the source WavPack file is being overwritten. Adding the -d option will accomplish this, however.

File Formats:

```
Microsoft Waveform Audio, extension ".wav", PCM audio
WavPack Compressed Audio, extension ".wv", PCM audio
Apple Core Audio Format, "extension ".caf", PCM or DSD audio
Sony Wave64 Audio Format, extension ".w64", PCM audio
Philips DSDIFF Format, "extension ".dff", 1-bit DSD audio
Sony DSD Stream Format, "extension ".dsf", 1-bit DSD audio
```

Both programs will warn before overwriting any file (unless the -y or --no-overwrite switch is specified) and file overwriting is implemented with temp files so that the overwritten file is not deleted until the operation is successful (including the verification pass if that has been requested with the -v switch). The programs will also (unless the -q switch is specified) display progress in percentage complete and when finished with each file will give the compression ratio (or bitrate for lossy files), whether the compression was lossy or lossless, and the processing time. They will also (unless the -z option is used) display the progress percentage for the whole batch of files on the title bar which is useful when they are being run minimized. If a batch of files is run then both programs will indicate when complete whether any errors occurred and how many files were processed.

The compressed file format (.wv) contains the filename extension, all the original header information, and the compressed audio data for one audio file. The file can be used by WvUnpack to restore the original file, or the .wv file can be played directly by a player that supports WavPack files natively like Foobar2000 or JRiver Media Center, or a player that can play WavPack files using a plugin like Winamp, dBpowerAMP, or Apollo. Also, there are plugins available to read and write WavPack files with Adobe Audition (and CoolEdit) and Nero Burning ROM. Many DAW programs also support WavPack natively like Reaper, Steinberg WaveLab, and Audacity.

Starting with version 5.5.0, the WavPack and WvUnpack programs allow options to be added to the executable (exe) filename and these options are parsed first every time the command is executed. This is useful when the programs are run by dropping files into their respective icons in the Windows File Explorer rather than using the Command Prompt to run them. For example, the WavPack executables might be renamed (see below for the exact meaning of the options listed):

```
wavpack{--pause}{--drop}{-hxvm}{--import-id3}.exe
wvunpack{--pause}{--drop}.exe
```

When multiple input files are specified for the WavPack and WvUnpack programs (using wildcards) and the output is specified as stdout ("-") then the resulting output from all the files processed is sent to stdout in a continuous concatenated stream. Normally this is not very useful because data is interleaved with file headers, however one case can be. If the --raw or --raw-pcm option is used in WvUnpack then just the audio from all the files will be send to stdout, and this can be captured in a file or piped directly back to WavPack for reëncoding:

```
wvunpack --raw *.wv - > merged.raw
wvunpack --raw *.wv - | wavpack --raw-pcm - merged.wv
```

Finally, a debugging mode is provided for the programs which is enabled by simply appending "_debug" to the the name of the executable (for example, wavpack.exe becomes wavpack_debug.exe). The debugging versions work exactly like the normal versions except that they report more detailed information about their operation. For instance, they will list their command-line arguments and wavpack.exe will display detailed information about the files it compresses (including how the audio data is interpreted). Also, all information displayed on the console (including any errors encountered) is dumped to a text file called "wavpack.log" (under Application Data). This is very handy for debugging in situations where the command-line window terminates before an error message can be read (or doesn't get displayed at all). It is not recommended that these debug version be used all the time because the log file will grow indefinitely.

Linux

This documentation was written specifically for the Windows command-line version of WavPack, however the version for Linux (and MacOS) is essentially the same. There are some Windows-specific commands missing from the Linux version like the -l switch to run at low priority or the --pause and --drop options helpful with drag and drop. The "debug" mode described above does not create a log file, but it *does* provide the extra information.

The most significant change is in the specification of the output filename or directory, which in the Linux case must be preceded with the -o switch. This was done to allow multiple input files to be specified, which is handy by itself but is especially useful in the default case where the shell is performing wildcard expansion.

WavPack Options

-a = Adobe Audition (CoolEdit) mode for 32-bit floats

The WAVEFORMATEXTENSIBLE structure is used (if present) to determine the format details. However, there are some programs that use their own non-standard format extensions. The most popular of these is Adobe's Audition (previously Syntrillium's CoolEdit) which created two new 32-bit floating point formats. An option has been added to WavPack (-a) to force the "adobe" interpretation of these floating point formats. If you are compressing integer files do NOT use this option.

--allow-huge-tags = allow tag data up to 16 MB (otherwise it's 1 MB)

Normally WavPack allows the APEv2 tags to contain up to 1 MB of data. This limit was implemented to allow for their use on portable devices which may have limited memory or processing resources. However, in some situations it may be desirable to place more data in the tags (for high resolution cover art scans, for example) and this option permits that. Note that these files are not fully WavPack compliant and may not work in all situations or with older versions of WavPack programs and plugins.

-bn = enable hybrid compression, n = 2.0 to 23.9 bits/sample, or
n = 24-9600 kbits/second (kbps)

The default operation of WavPack is pure lossless, which means that the .wv file contains all the information that was in the original .wav file. The hybrid mode allows the user to specify a target bitrate for the output file, either in kilobits per second (kbps) or bits per sample. If the track can be losslessly compressed without exceeding the specified bitrate, then it will be and WavPack will report the compression as lossless. If lossless compression would exceed the specified bitrate, then WavPack will begin carefully discarding the least significant portion of the audio information to stay within the limit. Every effort is made to keep this inaudible, including the use of joint stereo, dynamic bit allocation and noise shaping. WavPack will report this as "lossy" compression. Although the option accepts bitrates as low as 24 kbps, the actual value that WavPack can achieve is usually much higher than that. For example, with CD-audio sampled at 44.1k the lower limit is about 196 kbps.

The hybrid mode can be used quite successfully with floating-point audio, however it should **not** be used for scientific type floating-point data because the hybrid algorithm might not be application appropriate (and floating-point "exception" values like infinities or NaNs will not be properly encoded). Use only the pure lossless mode with non-audio floating-point data.

The hybrid mode is **not** usable with DSD audio files; those are always lossless and attempting hybrid compression with them will generate an error.

--blocksize=n = specify block size in samples (n = 128 - 131072)

WavPack normally determines the optimum number of samples to place into each WavPack block, however this option allows it to be directly specified. The most likely use for this option would be to improve the handling of audio files that have a variable number of redundant MSBs. For example, the output of the lossyWAV program or the output of a software HDCD decoder. This option is also often combined with the **--merge-blocks** option (which also reduces the minimum allowed block size to 16 samples).

-c = create correction file (.wvc) for hybrid mode (=lossless)

If the -c option is specified (in addition to the -b option), then WavPack will generate an additional file that will contain the information that was discarded to generate the lossy .wv file. This file will have the same name as the .wv file but will have the extension .wvc (the 'c' is for "correction"). When WvUnpack is used to restore the .wav file, it will attempt to find the .wvc file in the same directory as the .wv file. If it is found then the decompression will be lossless and will be reported as such, otherwise lossy decompression will be reported (assuming that any information was actually discarded during the original compression). If -c is specified but no actual information is discarded, the correction file will be deleted. The extra overhead involved with having these two files instead of a single pure lossless file is usually less than 1% of the original .WAV file and can be as low as 0.25% at high bitrates. Note that CRCs are stored for both the lossy and lossless versions, so error detection works correctly whether the .wvc file is used or not.

-cc = maximum hybrid compression (hurts lossy quality & decode speed)

Normally, when the -c option is used to create a correction file in the hybrid mode, WavPack attempts to optimize for the quality of the lossy file and lets the combined lossless compression of the two files fall where it may. This option tells WavPack to optimize for the overall compression ratio instead, even if this means some possible degradation of lossy quality (for example, dynamic noise shaping is not used in this mode). This can also have an effect on lossless decompression speed (however it does **not** affect lossy decoding). Keep in mind the effect of this option is not too significant either way.

--channel-order=<list> = specify non-standard channel ordering

For multichannel audio WAV files there is a Microsoft required WAVEFORMATEXTENSIBLE header to indicate which speakers are represented and those speakers must be in the standard Microsoft order (which is also specified by USB). However, some programs skip generating the

WAVEFORMATEXTENSIBLE header and even write the channels in the wrong order. For files without the WAVEFORMATEXTENSIBLE header (or those with the header but with a zeroed channel mask), WavPack assumes Microsoft channel order and further assumes that all speakers (up to the number of channels) are present. If that is not the case, then this option allows the user to specify the exact channels present (and their order) from this list (which is in Microsoft order):
FL,FR,FC,LFE,BL,BR,FLC,FRC,BC,SL,SR,TC,TFL,TFC,TFR,TBL,TBC,TBR.

If not all of the channels in the file have speaker definitions, then it is possible to simply terminate the speaker list with "..." (e.g. FL,FR,...) to indicate that all following channels are unassigned. Specifying "..." alone indicates that all channels are unassigned.

This option may just be used to specify *which* channels are present, even if they are in standard Microsoft order (e.g. standard quad is FL,FR,BL,BR). In this case the option just controls the channel "mask" and performs no reordering. But if the option does result in reordering, then this is done *before* encoding, so if the resulting WavPack file is subsequently unpacked it will **not** recreate the identical WAV file (because the channels will now be in the correct order). If you want the unpacked file to be identical to the original simply do not use this option (although this means that the resulting WavPack file will not have the correct channel information and so won't play correctly in multichannel software, and also means that the compression might not be as good because the "wrong" channels will be paired in stereo).

If this option is used then it might be a good idea to add the **-r** option also to generate a valid WAVEFORMATEXTENSIBLE header with the correct channel information. In fact, one could even pipe the output of WavPack directly to WvUnpack for the sole purpose of reordering WAV files and adding the correct WAV headers!

--cross-decorr = use cross-channel correlation in hybrid mode

Cross-channel correlation is exploited in lossless mode, but it not used by default in hybrid mode because it can increase noise slightly and increases CPU requirements during hybrid lossless decoding. This switch is provided to force this mode without affecting noise shaping (which **-cc** does to maximize hybrid lossless compression).

-d = delete source file if successful (use with caution!)

Self explanatory.

--drop = accept multiple files, no output specification allowed

This Windows-only option should be specified before any source files and alters the syntax to allow multiple input files instead of just one. This is useful for using the WavPack executable icon as the target of a "drag and drop" operation in which case the option is added to the executable filename using braces (e.g., **wavpack{--drop}.exe**). Because this is mostly intended for drap and drop support, the specification of output names or folders is not supported.

-f = fast mode (fast, but some compromise in compression ratio)

The "fast" mode should be used when compression (or decompression) speed is more important than compression ratio (or, in lossy mode, audio quality). This option has no effect on DSD audio.

-g = general/normal mode (cancels any previously specified -f or -h options)

This option is provided to revert to the normal default speed mode after having specified one of the other modes (i.e., "fast" or "high"). It really only makes sense if one of the other modes has been encoded directly into the executable filename using braces, but it could be specified to be unambiguous.

-h = high quality (better compression in all modes, but slower)

The "high" mode should be used when compression ratio (or, in lossy mode, audio quality) is more important than compression (or decompression) speed. This option slows both by about a factor of about 1.5.

-hh = very high quality (best compression in all modes, but slowest)

The "very high" mode should be used when compression ratio (or, in lossy mode, audio quality) is much more important than compression (or decompression) speed. This option slows both by about a factor of about 2, and is not recommended for use on vintage portable devices because of the high CPU load required for decoding, but should be fine for anything modern. This option simply activates the **high** mode for DSD audio files.

--help = extended help display

Self explanatory.

-i = ignore length in wav header (no pipe output allowed)

Some programs that pipe data to encoders do not always give the correct length in the wav headers that they provide (foobar's client and CDex are examples). In these cases use this option to force WavPack to ignore the header and accept the actual length. Because WavPack must seek to the beginning of the output file to write the correct length, this option cannot be used with piped output (override with **-y**). Note that WavPack will attempt to fix the supplied header with the correct length if it

can do so safely (and of course in this case it's not strictly lossless, but we're just fixing an invalid header). **Only use this option when it's really needed; using with a valid header will cause any trailing metadata to be interpreted as audio.**

When using **-i** it is not allowed to use pipes for both the input and output because the actual audio length will not be known in the beginning and there's no way to rewind the output file when we're done to update the length stored there. You can use the **-y** to override this but you will end up with sub-standard files that must be seeked to the end on decode to determine their length. A warning to this effect will be displayed when the operation completes.

--import-id3 = import applicable tag items from ID3v2.3 tag on DSF and other files

Sony's DSF file format specifies that these files may contain an ID3v2 tag at the end, and it is not uncommon for other files to contain these tags even though it's not part of their official specs (e.g., WAV and DSDIFF files). WavPack considers this a trailing "wrapper" and stores it in the WavPack file as such so that the original file can be restored verbatim. However, stored this way it is not easily accessible for reading (and it is certainly not writable) because WavPack uses APEv2 tags for metadata. This option causes any trailing ID3v2.3 tag in the DSF file (or any other file type) to be scanned and all applicable items imported into the APEv2 tag, including cover art.

Note that if over 1 MB of image data is present, then the **--allow-huge-tags** option must be specified. Also, keep in mind that the image data will be duplicated in the APEv2 tag and will therefore be consuming twice as much space as required, so it might make sense to also specify the **-r** option if very large images are present, although this will obviously make it impossible to restore the original DSF file because the ID3v2 tag will be gone, and of course there might be fields in the original tag that are not copied to the APEv2 tag.

-jn = joint-stereo override (0 = left/right, 1 = mid/side)

WavPack normally defaults to joint stereo (sometimes called mid/side) in which the left and right channels are combined to form an alternate representation (essentially L+R and L-R) that compresses better in lossless mode and improves quality in lossy mode. In the "extra" modes WavPack will choose whether or not to use joint-stereo on a block by block basis. This option allows this feature to be forced either always on or always off.

-l = run at low priority (for smoother multitasking)

This option can be used (in Windows only) to force WavPack to run at a low priority and is handy for doing large WavPack batch conversions in the background.

-m = compute & store MD5 signature of raw audio data

Calculate and display the MD5 checksum of the uncompressed audio data and store it in the compressed file. These sums are commonly used in file trading communities to compare versions of tracks, and as such the sums generated by WavPack match those of FLAC, OptimFROG, Shntool, and get_id3(). They can also be used by WvUnpack during decompression to verify the data integrity of lossless files, but this is a secondary function because WavPack already verifies each block as they are decoded.

--merge-blocks = merge consecutive blocks of equal redundancy

This option is only valid when generating lossless files and is only used with the **--blocksize** option. WavPack will always scan each block of audio data before compressing to determine the actual number of valid bits that need to be stored because, for various reasons, this is not always the same as the number of bits present. In some situations there can be many redundant bits, as for example in the output of the lossyWAV program or the output of software HDCD decoders. In these cases it may be advantageous to use a specific block size (or simply a small block size), but WavPack does not work very well with very small block sizes because of its relatively large block header.

The **--merge-blocks** option takes care of this situation by allowing WavPack to merge consecutive blocks with the same number of bits to remove. For example, if a blocksize of 512 is specified, and 10 512-sample blocks in a row all have exactly the same number of redundant bits, then these 10 blocks will be combined in to a single block of 5120 samples (which is much more efficient for WavPack). When the **--merge-blocks** option is specified the minimum block size that **--blocksize** will accept is reduced from 128 to 16 samples.

-n = calculate average and peak quantization noise (hybrid only)

This causes WavPack to calculate the average and peak quantization noise generated in the lossy version of the hybrid mode, both referenced in decibels below full scale. While it is impossible to use this as a guide to determine the audibility of the noise, it is useful for comparing the various compression options and for comparing WavPack's lossy performance to other programs. Note that this option does not currently produce meaningful results for floating point or multichannel files.

--no-overwrite = don't overwrite existing files, and don't ask to

Normally WavPack checks that a file it is about to create already exists, and if it does will prompt the user before overwriting it. When this option is specified WavPack will skip this prompt and simply go to the next file (if there are any). This might be useful to resume a long, multi-file WavPack operation that

was cancelled because in that case all the previously encoded files are quickly skipped.

--no-utf8-convert = don't recode passed tags to UTF-8, assume they are UTF-8 already

The text fields of APEv2 tags are encoded in the UTF-8 variant of Unicode, so when tag information is passed in on the command-line they are converted to UTF-8 before being stored. If your system is already passing the strings in UTF-8, use this option to prevent double conversion.

--pair-unassigned-chans = encode unassigned channels into stereo pairs

WavPack encodes all channels of multichannel files into stereo or mono streams. So, if two channels are related (like FL and FR) then these are encoded into one stream (which improves compression performance) whereas unrelated channels (like LFE) are encoded by themselves into mono streams. Unassigned channels (those not associated with any defined speaker) are normally encoded in mono streams for safety, however this option can be used to specify that they should be paired into stereo streams. This might improve compression in cases where all of the channels were very similar, or when it is known that the channels are, in fact, stereo pairs.

--pause

Pause before exiting the console program (Windows only), allowing the user to press any key to continue. This might be useful to include in situations where the console window disappears before the completion status can be seen (like with EAC or the WavPack FrontEnd) or when using the program icon as "drag and drop".

--pre-quantize=bits = apply pre-quantization to specified bitdepth

Even the finest 24-bits ADCs have a true resolution of only around 20 or 21 bits, and those lower "noise" bits are completely uncompressible. This option enables truncation of the source audio samples to a specified bit depth **before** encoding and MD5 calculation, and can often greatly improve the lossless compression ratio. Keep in mind that although this is conceptually a lossy operation, it will be reported as lossless because this is performed by the command-line program itself just as the audio is read from the source file (but of course the source file is not actually modified).

This option can be used with any source format (float data is rounded instead of truncated, and *not* clipped) and any WavPack mode, although it really does not make sense to use this in combination with WavPack's lossy mode because it wouldn't do much (at least not with reasonable parameters). In the hybrid lossless mode, this pre-quantizing *would* reduce the size of the correction file, but wouldn't have much effect on the lossy portion. It also works well as a reëncoding operation (with a WavPack source file), although it *will* modify the MD5 checksum stored in the file (which WavPack's lossy mode does not do).

The pre-quantize option is **not** usable with DSD audio files because they are already only 1 bit deep, but an error will not be generated (the option is ignored).

-q = quiet (keep console output to a minimum)

Self explanatory.

-r = remove any extra chunk info from audio file

WavPack normally saves all the header information contained in the audio file (including any chunks after the audio data). This is done so that WvUnpack.exe can restore the original audio file *exactly*. The -r option causes WavPack to discard the header contained in the source file (and any extra chunks). Obviously the source header is still used to determine the format and size of the file (i.e. this is not a "raw" mode). When the file is restored with WvUnpack, a generic header will be generated appropriate for the format. Note that if this option is used with certain floating-point WAV files generated by CoolEdit or Audition that are not normalized, the **--normalize-floats** option of WvUnpack will have to be used when converting them back to standard PCM formats.

--raw-pcm = input data is raw pcm (44100 Hz, 16-bit signed, 2-ch, little-endian)

Specifies that the source file (or stdin) contains raw PCM data with no header. and conforms to standard CD quality. Note that the data must match the format of WAV file audio (i.e., little endian and signed).

When encoding raw PCM, it is not allowed to use pipes for both the input and output because the actual audio length will not be known in the beginning and there's no way to rewind the output file when we're done to update the length stored there. You can use the **-y** to override this but you will end up with sub-standard files that must be seeked to the end on decode to determine their length. A warning to this effect will be displayed when the operation completes.

Note that WavPack files created in this way will still normally decode to WAV files (with headers) when unpacked. To avoid this, use the **-r**, **--raw**, or **--raw-pcm** option with the WvUnpack program to force raw unpacking.

--raw-pcm=sr,bps[f|s|u],ch,[le|be] = input data is raw pcm with specified format

Specifies that the source file (or stdin) contains raw PCM data with no header and the specified sampling rate, bit-depth, and number of channels. Parameters matching the default (44100,16,2,le) can be omitted. The valid range of bit-depths is from 1 to 32. The default data format can be overridden with

"f" for floating-point audio (32-bit only) or "s" or "u" for signed/unsigned integers. The number of channels may be from 1 to 4096.

To compress DSD audio, specify bps=1. Note that DSD audio must be in Philips DSDIFF format (i.e., channels interleaved by byte and MSB first temporally). Therefore, standard SACD stereo audio would be **--raw-pcm=2822400,1,2**.

Note that WavPack files created in this way will still normally decode to WAV files (with headers) when unpacked (or DFF files for DSD audio). To avoid this, use the **-r** option with the WvUnpack program to force raw unpacking.

-sn = noise shaping override (hybrid only, n = -1.0 to 1.0, 0 = off)

WavPack uses first-order noise shaping to improve the perceived quality of lossy files and to improve the hybrid lossless compression ratio. Normally WavPack will choose the noise shaping most appropriate for the application (based on the source sampling rate and whether the -cc option is specified), but this option allows the user to override the default with a fixed value. The parameter range is +/- 1.0, where positive values shift the noise higher in frequency and negative values shift the noise lower in frequency. Values very close to -1.0 are clipped to prevent problems associated with very high gain near DC and the value zero results in no noise shaping at all (i.e., white noise). This option should not be used with the **--use-dns** option which enables dynamic noise shaping.

-t = copy input file's time stamp to output file(s)

Self explanatory.

--use-dns = force use of dynamic noise shaping (hybrid only)

For version 4.50, dynamic noise shaping was added to the WavPack hybrid mode. This feature causes the noise shaping to continuously adjust to the spectral characteristics of the source signal and results in significantly improved subjective quality on samples that previously caused difficulty with WavPack's lossy mode (particularly material with loud high-frequency transients). This feature is used by default except for high sampling rates (>= 64 kHz) or when the -cc option is specified (because it can adversely effect compression ratio), however the **--use-dns** option allows the user to force the use of dynamic noise shaping even when it would not normally be used. This option should not be specified with the **-s** option which overrides the default with fixed noise shaping.

-v = verify output file integrity after write

Causes WavPack to perform a separate verification pass over the output file to guarantee that the audio data is correctly encoded and stored to disk. For lossless compression this is performed with an MD5 hash even if the **-m** option is not specified. If an error occurs (indicating either a bug in the program or faulty hardware) then a message is displayed and the output file is deleted (and if an existing file is being overwritten, then it will be untouched). Because the output file is actually rewound and reread, this option cannot be used when writing to pipes.

--version = display program version to stdout

Both the version of the command-line program and the WavPack library are displayed.

-w Encoder = write actual encoder metadata to APEv2 tag

Write a metadata item to the APEv2 tag for the actual encoder being used (e.g., "Encoder=WavPack 5.5.0"). It's also possible to specify a value to override the default. Note that when reëncoding WavPack files this metadata item is **not** simply copied from source to destination, but is recreated with the appropriate value.

-w Settings = write actual encoder settings metadata to APEv2 tag

Write a metadata item to the APEv2 tag for the actual encoder settings being used (e.g., "Settings=-hb384cx3"). It's also possible to specify a value to override the default. Note that when reëncoding WavPack files this metadata item is **not** simply copied from source to destination, but is recreated with the appropriate value.

-w "Field=[@]Value" = write specified metadata to APEv2 tag

Write specified information to APEv2 tag appended to WavPack file(s). May be used multiple times for multiple fields. APEv2 tags are the preferred tag format for WavPack files and are read by all the standard WavPack playback plugins. If the specified value begins with a '@', then the value is assumed to be a filename which is used to obtain the item's actual value. This is handy for including the CUESHEET field for use with images files + cuesheets and foobar2000. The filename may contain wildcards if it matches exactly one file. Also, if the file cannot be found in the current directory then the source and destination directories (if specified) are also checked.

--write-binary-tag "Field=@file.ext" = write specified binary metadata to APEv2 tag

Write specified binary file to APEv2 tag appended to WavPack file(s). This is most commonly used to embed album cover art into WavPack files (with the field name **"Cover Art (Front)"**), but could be used for anything desired. A file must be specified (the data cannot come from the command-line) and it may contain wildcards if it matches exactly one file. Also, if the file cannot be found in the current

directory then the source and destination directories (if specified) are also checked.

-x[n] = extra encode processing (optional n = 0-6, 1 = default)

Like pre-4.0 versions of WavPack (and many other compressors), WavPack 5.5.0 normally works "symmetrically" in that encoding and decoding operate at about the same rate (regardless of the mode used). However, WavPack has an option to work "asymmetrically", so that extra processing can be done during encoding to provide better compression, but with NO corresponding cost to decoding performance!

This is enabled with the -x option and provides an average improvement in CD music compression of about 1% in "fast" mode, about 0.5% in the normal mode, and still less in the higher modes. Because the standard compression parameters are optimized for "normal" CD music audio, this option works best with "non-standard" audio (synthesized sounds, non-standard sampling rates, etc.) where it can often achieve enormous gains. The default level (n=1) provides a decent improvement with little cost in encoding speed and is recommended for all but the most time critical encoding. Higher levels provide some marginal improvement with an increasing cost of encoding speed. The highest levels (n = 4-6) are extremely slow but can provide significant improvement in special situations (i.e. synthesized sounds). Use -x0 to specify no extra processing.

This option is not applicable to DSD audio and is simply ignored.

-y = yes to all warnings (use with caution!)

Self explanatory.

-z[n] = don't (n=0 or omit) or do (n=1) write to console title bar to show overall progress

If enabled leaves "WavPack Completed" on the title bar.

WvUnpack Options

-b = blindly decode all stream blocks & ignore length info

This option uses the "streaming" mode of the WavPack decoder. Normally this doesn't make any difference except that the percentage complete display does not work. However, it causes the decoder to ignore all information contained in the WavPack blocks concerned with duration and location and blindly unpacks every valid WavPack block it encounters, and will do so indefinitely until it receives an EOF. This may be of some use to recover corrupted or partial WavPack files, or it could be used to decode a sequence of WavPack files that had been concatenated. If this is used to write a .wav file to stdout then note that the resulting .wav file will not indicate the correct length because it can't back up to fix it once it knows the actual length.

-c = extract cuesheet only to stdout (no audio decode)

If the specified WavPack file contains an APEv2 tag, and that tag contains a CUESHEET field, then only dump that text to stdout without decoding any audio. If no cuesheet is found then an error is generated. This is equivalent to "-x cuesheet".

-cc = extract cuesheet file (.cue) in addition to audio file

If the specified WavPack file contains an APEv2 tag, and that tag contains a CUESHEET field, then that text will be extracted into an appropriately named .cue file located in the same folder as the .wav file. Note that the cuesheet is not automatically modified to point to the correct .wav file; this must be handled either when the cuesheet is embedded or (if the .wv file is renamed) by the user after extraction. This option has no effect if there's no cuesheet and is equivalent to "-xx cuesheet=%a.cue".

--caf-be or --caf-le = force output to Apple Core Audio format

Output an Apple Core Audio file (either big- or little-endian) regardless of the input file format. All extra information in the original file's header and trailer will be lost and a "fresh" Core Audio header will be generated. Note that DSD audio files will be decimated 8X and output as 24-bit PCM.

-d = delete source file if successful (use with caution!)

Self explanatory.

--dff or --dsdiff = force output to Philips DSDIFF format

Output a Philips DSDIFF file (extension .dff) audio file regardless of the input file format. All extra information in the original file's header and trailer will be lost and a "fresh" DSDIFF header will be generated. Note that only DSD audio files can be exported as DSDIFF files (WvUnpack does not contain a DSD encoder).

--drop = accept multiple files, no output specification allowed

This option should be specified before any source files and alters the syntax to allow multiple input files instead of just one. This is useful for using the WvUnpack executable icon as the target of a "drag and drop" operation in which case the option is added to the executable filename using braces (e.g., **wvunpack{--drop}.exe**). Because this is mostly intended for drap and drop support, the specification of output names or folders is not supported.

--dsf = force output to Sony DSD format

Output a Sony DSD Stream File (extension .dsf) regardless of the input file format. All extra information in the original file's header and trailer will be lost and a "fresh" DSF header will be generated. Note that only DSD audio files can be exported as DSF files (WvUnpack does not contain a DSD encoder).

-f[n] = dump summary file information only to stdout in machine-parsable format (no decode)

This command would normally be used by another application to obtain information about WavPack files. It outputs a single, semicolon delimited line to stdout for each input file specified. Here is a sample output line and a description of the 10 items of information provided:

44100;16;int;2;0x3;9878400;023066a6345773674c0755ee6be54d87;4;0x18a2;Track01.wv

1. sampling rate
2. bit-depth (1-32)
3. format ("int" or "float")
4. number of channels
5. channel mask (in hex because it's a mask, always prefixed with "0x")
6. number of samples (missing if unknown)
7. md5sum (technically is hex, but not prefixed with "0x", might be missing)
8. encoder version (this will be 4 or 5 as older files require a specially built decoder)
9. encoding mode (in hex because it's a bitfield, always prefixed with "0x")
10. filename (if available)

The optional parameter can be used to output a single item from the 10 available.

--help = this help display

Self explanatory.

-i = ignore .wvc file (forces hybrid lossy decompression)

This option forces WvUnpack to *not* look for a corresponding correction file when unpacking hybrid mode files. This could be used to compare the effects of the hybrid lossy mode or in the case that the .wvc had become corrupt and was preventing decompression (although this is unlikely).

-l = run at low priority (for smoother multitasking)

This option can be used (in Windows only) to force WavPack to run at a low priority and is handy for doing large unpack or verification batch conversions in the background.

-m = calculate and display MD5 signature; verify if lossless

Enables the calculation and display of the MD5 checksum on the uncompressed audio data. If an MD5 sum for the original audio data is stored in the WavPack file then this will be displayed also. For lossless operation these numbers should match (and WvUnpack will verify that they match). For lossy operation these numbers should not match.

-n = no audio decoding

Use this option with the **-xx** and **-cc** options to perform the tag extraction only.

--normalize-floats = normalize floating-point audio if it isn't already

Floating-point audio is considered "normalized" when the fullscale range is +/-1.0 (although peaks may exceed this range, which is one of its advantages). Most PCM file formats (WAV, CAF, etc) require normalized floating-point audio, but it's possible to have WavPack floating-point audio files that are normalized to a different range, specifically those created from certain Cool Edit or Adobe Audition versions. If it is desired to export such files to a standard PCM format (or a normalized raw file) then include this option. Note that since the audio will be scaled, the operation will no longer be strictly lossless and so the option for verifying the MD5 checksum should not be used.

--no-utf8-convert = leave tag items in UTF-8 on extract or display

The text fields of APEv2 tags are encoded in the UTF-8 variant of Unicode, so when tag information is displayed or extracted, it is converted to local multibyte encoding before being displayed or written to files. If your system already expects UTF-8, use this option to prevent the conversion.

--pause

Pause before exiting the console program (Windows only), allowing the user to press any key to continue. This might be useful to include in situations where the console window disappears before the completion status can be seen (like the WavPack FrontEnd) or when using the program icon as "drag and drop".

-q = quiet (keep console output to a minimum)

Self explanatory.

-r or --raw = force raw audio decode (results in .raw extension)

This option causes WvUnpack to strip off any file header (and any trailer) and only output the uncompressed audio data. If a file is created then it will be given the .raw extension instead of the extension on the original file. Note that the raw data still conforms to standard WAV audio data (i.e., little-endian and signed, except for 8-bit data which is unsigned). DSD audio data is output in Philips DSDIFF format (i.e., channels interleaved by bytes, MSB first temporally) unless the **--raw-pcm** option is used instead.

--raw-pcm = force raw PCM audio decode (results in .raw extension)

This option is identical to the **-r** and **--raw** option above except in the case of DSD-encoded WavPack files which are decimated 8x to 24-bit PCM instead of being output in raw DSD. Note that the decimated PCM audio still contains large amounts of DSD quantization noise above the audible range and should be filtered or downsampled appropriately.

-s = display summary information only to stdout (no decode)

This option causes WvUnpack to simply display general information about the specified WavPack files including source information, encoder version and options and original MD5 checksum. This information can be directed into a file with the ">" operator. No file decoding or verification can occur with this option.

-ss = display super summary (including tags) to stdout (no decode)

This option is similar to -s except that information from any valid tag found (either APEv2 or ID3v1) is also listed. For binary or multi-line text items, only the size of the tag data is displayed, but you can use the -x command to view this data (or redirect it to a file).

--skip=[-][sample|hh:mm:ss.ss] = start decoding at specified sample/time

Specifies an alternate start position for decoding, as either an integer sample index or as a time in

hours, minutes, and seconds (with fraction). A minus ('-') sign indicates that the time or sample index is relative to the end of the file. The WavPack file must be seekable (i.e. not a pipe). This option can be used with the --until option to decode a specific region of a track.

-t = copy input file's time stamp to output file(s)

Self explanatory.

--until=[+|-][sample|hh:mm:ss.ss] = stop decoding at specified sample/time

Specifies an alternate stop position for decoding, as either an integer sample index or as a time in hours, minutes, and seconds (with fraction). If a plus ('+') or minus ('-') sign is inserted before the specified sample (or time) then it becomes a relative amount, either from the position specified by a --start option (if plus) or from the end of the file (if minus).

-v = verify source data only (no output file created)

This option allows the user to verify the integrity of WavPack files (including any correction file). Note that this option not only verifies that a WavPack file has not been corrupted since creation, but it also verifies that the audio data is being unpacked exactly as intended (even in the lossy mode). Therefore, it can also detect algorithm errors, incompatible implementations of the encoder or decoder, and even faulty processor hardware.

-vv = quick verify (no output, version 5+ files only)

WavPack blocks have always contained a checksum for the audio data that is used during decoding (or verifying) to make sure the block is not corrupt. Starting with version 5 however, WavPack blocks contain an additional checksum for the *encoded* data block which is used to increase robustness during decode by rejecting corrupt blocks before they are even parsed.

This option allows the user to verify the blocks in a WavPack file using just these additional checksums, and because the audio is not decoded this operation is *much* faster than the regular verify option (-v). It is not as complete as the full verify for detecting all possible issues, but should easily detect the case where a WavPack file has been corrupted due to a transfer or storage problem. This operation will automatically fall back to the regular (slow) verify mode if older files that do not have the new checksum are encountered.

-vvv = quick verify verbose

This is similar to the -vv option above but also displays additional information that may be useful or interesting, like the channel count or chunks of unknown data between the WavPack blocks (which is valid up to 1 MB). This will also display whether the total length of the WavPack file is missing from the first block (which is valid, but means that the file requires an extra seek to the end to determine the length).

--version = display program version to stdout

Both the version of the command-line program and the WavPack library are displayed.

-w or --wav = force output to Microsoft WAV format

Output a Microsoft WAV file regardless of the input file format. All extra information in the original file's header and trailer will be lost and a "fresh" WAV header will be generated. For multichannel files, a WAVEFORMATEXTENSIBLE header is written, and for files over 4 GB, an RF64 file will be written (still with .wav extension). Note that DSD audio files will be decimated 8X and output as 24-bit PCM.

--w64 = force output to Sony Wave64 format

Output a Sony Wave64 file regardless of the input file format. All extra information in the original file's header and trailer will be lost and a "fresh" Wave64 header will be generated. Note that DSD audio files will be decimated 8X and output as 24-bit PCM.

-x "Field" = extract specified tag field only to stdout (no audio decode)

If the specified WavPack file contains a tag, and that tag contains the specified field, then only dump that field to stdout without decoding any audio. If the tag field is not found then an error is generated. Of course, the data may be redirected to a file or pipe. Only one field may be specified, and -c can be used as a shortcut to extract the CUESHEET field.

-xx "field[=file]" = extract specified tag field to file (with audio decode)

If the specified WavPack file contains a tag, and that tag contains the specified field, then that field will be extracted into an appropriately named file located in the same folder as the target .wav file. The name of the file will be the name of the tag field with a .txt extension for text items or, in the case of binary items, the name comes from the tag itself (although the convention is for this to be the tag field name with the extension from the original source file).

If the automatically generated name is not acceptable, then a new name can be specified, and this new name specification may contain the following replacement codes:

%a = audio output filename
%t = tag field name (note: comes from data for binary tags)
%e = extension from binary tag source file, or 'txt' for text tag

-y = yes to overwrite warning (use with caution!)

Self explanatory.

-z[n] = don't (n=0 or omitted) or do (n=1) write to console title bar to show overall progress

If enabled leaves "WvUnpack Completed" on the title bar.

WvGain Options

-a = album mode (all files scanned are considered an album)

The default mode for WvGain is to operate in *track* mode where each WavPack file is independently analyzed for perceived volume (and peak level). This option causes WvGain to additionally operate in *album* mode where all the specified files are analyzed together as an "album" so that a set of composite values are also generated. Note that when this mode is specified, WvGain must wait until the end of scanning before making another pass through all the files to append the calculated values.

-c = clean ReplayGain values from all files (no analysis)

This option does not perform any analysis but simply removes all ReplayGain values from the specified files.

-d = display calculated values only (no files are modified)

This option causes WvGain to do the specified analysis (with or with the -a switch for *album* mode) and display the calculated values, but no values are actually written to the files.

-i = ignore .wvc file (forces hybrid lossy decompression)

This option forces WvGain to not look for a corresponding correction file when analyzing hybrid lossless mode files. This is probably a good idea in most cases because it's faster and the lossy versions tend to have slightly higher peak amplitudes than the lossless versions (although the overall levels will be almost identical).

-l = run at low priority (for smoother multitasking)

This option can be used (in Windows only) to force WvGain to run at a low priority and is handy for doing large batch operations in the background.

-n = new files only

This option instructs WvGain to check the tags of the files first to make sure that they don't already have ReplayGain information. Files that already have ReplayGain information are skipped; in "album" mode a file encountered with ReplayGain information will abort the whole operation.

-q = quiet (keep console output to a minimum)

Self explanatory.

-s = show stored values only (no analysis)

This option does not perform any analysis but simply displays (to stdout) any ReplayGain values stored in the specified WavPack files.

-v = display program version to stdout

Both the version of the command-line program and the WavPack library are displayed.

-z[n] = don't (n=0 or omit) or do (n=1) write to console title bar to show overall progress

If enabled leaves "WvGain Completed" on the title bar.

WvTag Options

--allow-huge-tags = allow tag data up to 16 MB (otherwise it's 1 MB)

Normally WavPack allows the APEv2 tags to contain up to 1 MB of data. This limit was implemented to allow for their use on portable devices which may have limited memory or processing resources. However, in some situations it may be desirable to place more data in the tags (for high resolution cover art scans, for example) and this option permits that. Note that these files are not fully WavPack compliant and may not work in all situations or with older versions of WavPack programs and plugins.

-c = extract cuesheet only to stdout

If the specified WavPack file contains an APEv2 tag, and that tag contains a CUESHEET field, then dump that text to stdout. If no cuesheet is found then an error is generated. This is equivalent to "-x cuesheet".

-cc = extract cuesheet file (.cue)

If the specified WavPack file contains an APEv2 tag, and that tag contains a CUESHEET field, then that text will be extracted into an appropriately named .cue file located in the same folder as the source file. This option has no effect if there's no cuesheet and is equivalent to "-xx cuesheet=%a.cue".

--clean or --clear = clean all tag items from APEv2 tag (done first)

This option, if specified, is performed first and results in a fresh tag to start with.

-d "Field" = delete specified metadata item (either text or binary) from APEv2 tag

Self explanatory.

-h or --help = display usage information

Self explanatory.

--import-id3 = import applicable tag items from ID3v2.3 tag on DSF and other files

Sony's DSF file format specifies that these files may contain an ID3v2 tag at the end. WavPack considers this a trailing "wrapper" and stores it in the WavPack file as such so that the DSF file can be restored verbatim. However, stored this way it is not easily accessible for reading (and it is certainly not writable) because WavPack uses APEv2 (or, sometimes, ID3v1) tags for metadata. This option causes any trailing ID3v2.3 tag in the DSF file (or other file type) to be scanned and all applicable items imported into the APEv2 tag, including cover art.

Note that if over 1 MB of image data is present, then the **--allow-huge-tags** option must be included

-l or --list = list all tag items (done last)

Self explanatory.

--no-utf8-convert = don't recode passed tags to UTF-8, assume they are UTF-8 already

The text fields of APEv2 tags are encoded in the UTF-8 variant of Unicode, so when tag information is passed in on the command-line they are converted to UTF-8 before being stored. If your system is already passing the strings in UTF-8, use this option to prevent double conversion.

-q = quiet (keep console output to a minimum)

Self explanatory.

-v or --version = display program version to stdout

Both the version of the command-line program and the WavPack library are displayed.

-w "Field=[@]Value" = write specified metadata to APEv2 tag

Write specified information to APEv2 tag appended to WavPack file(s). May be used multiple times for multiple fields. APEv2 tags are the preferred tag format for WavPack files and are read by all the standard WavPack playback plugins. If the specified value begins with a '@', then the value is assumed to be a filename which is used to obtain the item's actual value. This is handy for including the CUESHEET field for use with images files + cuesheets and foobar2000. The filename may contain wildcards if it matches exactly one file. Also, if the file cannot be found in the current directory then the source directory (if specified) is also checked.

--write-binary-tag "Field=@file.ext" = write specified binary metadata to APEv2 tag

Write specified binary file to APEv2 tag appended to WavPack file(s). This is most commonly used to embed album cover art into WavPack files (with the field name **"Cover Art (Front)"**), but could be used for anything desired. A file must be specified (the data cannot come from the command-line) and it may contain wildcards if it matches exactly one file. Also, if the file cannot be found in the current directory then the source directory (if specified) is also checked.

-x "Field" = extract specified tag field only to stdout (no audio decode)

If the specified WavPack file contains a tag, and that tag contains the specified field, then dump that field to stdout. If the tag field is not found then an error is generated. Of course, the data may be redirected to a file or pipe. Only one field may be specified, and -c can be used as a shortcut to extract

the CUESHEET field.

-xx "field[=file]" = extract specified tag field to file (with audio decode)

If the specified WavPack file contains a tag, and that tag contains the specified field, then that field will be extracted into an appropriately named file located in the same folder as the source file. The name of the file will be the name of the tag field with a .txt extension for text items or, in the case of binary items, the name comes from the tag itself (although the convention is for this to be the tag field name with the extension from the original source file).

If the automatically generated name is not acceptable, then a new name can be specified, and this new name specification may contain the following replacement codes:

%a = audio output filename
%t = tag field name (note: comes from data for binary tags)
%e = extension from binary tag source file, or 'txt' for text tag

-y = yes to overwrite warning (use with caution!)

Self explanatory.

Usage Guide

The options of WavPack can be a little confusing at first, so here's a little tutorial on what to try first if you're not sure. First of all, if you are only interested in lossless compression, try the default operation with no options. This will give a decent compression ratio at a very good speed for both packing and unpacking. Also, it is usually recommended to add the -x switch because it can provide a decent improvement in compression with only a minor cost in encoding time (about 2x) and *no* cost in decoding speed.

If speed is less of an issue try the "high" mode (-h), this will increase both the packing and unpacking time by about 50% (although still not nearly as slow as many other lossless compressors) while improving the compression ratio somewhat. If speed is paramount use the "fast" (-f) option. This gives the fastest packing and unpacking speed possible with WavPack, while still providing a very reasonable compression ratio. Again, the -x switch works well with these two modes.

If decoding speed is important, but the time to encode is not (because it happens only once), try using numeric parameters with the -x mode (valid values are 2-6). This will slow the encoding way down (depending on the level) but still have no effect on decoding speed. In some music files, this option can actually bump the compression ratio a whole mode (i.e. the "fast" mode can match the default mode, etc.). For the best lossless compression WavPack can offer, use -hhx6 (but be prepared to make yourself a sandwich!).

If you want to try the hybrid mode, all the above applies but you also have to choose a bitrate. The quality of WavPack's lossy mode cannot match the conventional lossy codecs like MP3 and WMA at similar bitrates, and in fact it won't even operate at the most common bitrate of 128 kbps (with CD audio, at least). The lowest bitrate that I recommend for WavPack lossy is 256 kbps which can provide transparent reproduction for most non-critical listening situations and is roughly equivalent to LAME MP3 encoding somewhere between 160 kbps and 192 kbps (CBR). Above 256 kbps the quality of WavPack's lossy mode increases rapidly, with added quantization noise (which is the only artifact) dropping by about 1 dB for every 15 kbps. At 320 kbps the quality is difficult for even critical listeners to distinguish from the original, and at 384 kbps WavPack becomes essentially transparent.

The various modes and options that apply to lossless mode also apply to hybrid mode. However the "extra" mode (-x) deserves a special mention. Most of the time, the "extra" mode makes only modest improvements because WavPack has already been finely tuned for ordinary music. However, sometimes there is an instrument or sound situation that does not compress well with the standard settings and the "extra" mode can make a significant improvement. In lossless mode this would slightly improve the overall compression ratio and would probably go unnoticed. However, in lossy mode that difficult section might trigger clearly audible noise to be added, and in this case the "extra" mode would save the day by greatly reducing the noise in the exact spot where it might have been audible.

Here is a chart of recommended settings for the various useful bitrates:

Bitrate	High quality	Faster Encode	Faster Decode
256 kbps	-hb256x3	-hb256x	-b256x3
320 kbps	-hb320x3	-hb320x	-b320x3
384 kbps	-b384x3	-hb384x	-fb384x3

Non-CD

In addition to regular CD audio data, WavPack hybrid mode is also well suited to other sampling rates, bitdepths, and channel configurations. For generating bitrates lower than the minimum of 196 kbps that applies to CD audio, it is necessary to resample the audio to lower sampling rates and/or convert to mono using an external program. Note that conventional lossy codecs accomplish the same thing by lowpass filtering the source when used for low bitrates.

For high-resolution audio (e.g., 24-bit/96-kHz) WavPack hybrid makes a particularly logical choice. The reasoning here is that standard lossless encoding achieves relatively poor compression on this type of material because the additional 8 bits are essentially noise and cannot be further compressed. In addition, conventional lossy codecs like MP3 or AAC are based on psychoacoustic models of hearing that presume CD audio quality (16-bit/44-kHz) to be "perfect", and therefore discard everything outside of that range.

When used at around 1024 kbps, WavPack hybrid preserves the full bandwidth and dynamic range of 24/96 sources. It accomplishes this by moving most of the quantization noise up above the audible audio band using first-order noise shaping in much the same way that DSD does. In fact, at normally loud listening levels, the added quantization noise remains completely inaudible, *even without the masking effect of the material itself!* While 1024 kbps may seem high at first, keep in mind that this is right in the range of some losslessly compressed CD-quality music, and is approximately 1/3 of the bitrate required to losslessly compress the same 24/96 material!

WavPack also handles multichannel audio like 5.1 and 7.1 channel configurations, and these can be stored in lossless and hybrid modes (including hybrid lossless). With all the combinations of sampling rates and numbers of channels it might become confusing to determine a desirable bitrate. This is an ideal application of the alternate form of the **-b** option which accepts a number of bits per sample rather than kilobits per second. When using this form a reasonable starting point of experimentation is 4 bits per sample (-b4). Except for extreme cases this will produce transparent results (it is equivalent to about 350 kbps for CD audio). Note that the compression algorithms are identical when using this form; it is simply an alternate way of specifying the bitrate.

Representative samples ranging from 24 kbps to 1024 kbps are found in the "hybrid_bitrates" directory of the WavPack Test Suite which is currently hosted here, but which can always be found on the WavPack website.

Plugins

The documentation for the various WavPack plugins is provided as readme.txt files included with those plugins.