FLAC Logo

home | faq | documentation | developers | changelog | more

flac

Table of Contents

General Usage

flac is the command-line file encoder/decoder. The encoder currently supports as input RIFF WAVE, Wave64, RF64, AIFF, FLAC or Ogg FLAC format, or raw interleaved samples. The decoder currently can output to RIFF WAVE, Wave64, RF64, or AIFF format, or raw interleaved samples. flac only supports linear PCM samples (in other words, no A-LAW, uLAW, etc.), and the input must be between 4 and 24 bits per sample. This is not a limitation of the FLAC format, just the reference encoder/decoder.

flac assumes that files ending in ".wav" or that have the RIFF WAVE header present are WAVE files, files ending in ".w64" or have the Wave64 header present are Wave64 files, files ending in ".rf64" or have the RF64 header present are RF64 files, files ending in ".aif" or ".aiff" or have the AIFF header present are AIFF files, and files ending in ".flac" or have the FLAC header present are FLAC files. This assumption may be overridden with a command-line option. It also assumes that files ending in ".oga" or ".ogg" or have the Ogg FLAC header present are Ogg FLAC files. Other than this, flac makes no assumptions about file extensions, though the convention is that FLAC files have the extension ".flac" (or ".fla" on ancient "8.3" file systems like FAT-16).

Before going into the full command-line description, a few other things help to sort it out: 1) flac encodes by default, so you must use **-d** to decode; 2) the options -0 .. -8 (or --fast and --best) that control the compression level actually are just synonyms for different groups of specific encoding options (described later) and you can get the same effect by using the same options; 3) flac behaves similarly to gzip in the way it handles input and output files.

Skip to the tutorial below for examples of some common tasks.

flac will be invoked one of four ways, depending on whether you are encoding, decoding, testing, or analyzing:

- Encoding: flac [*<general-options>*] [*<format-options>*] [*<encoding options>*] [inputfile [...]]
- Decoding: flac -d [*<general-options>*] [*<format-options>*] [*<decoding options>*] [FLACfile [...]]
- Testing: flac -t [*<general-options>*] [FLACfile [...]]
- Analyzing: flac -a [*<general-options>*] [*<analysis-options>*] [FLACfile [...]]

In any case, if no inputfile is specified, stdin is assumed. If only one inputfile is specified, it may be "-" for stdin. When stdin is used as input, flac will write to stdout. Otherwise flac will perform the desired operation on each input file to similarly named output files (meaning for encoding, the extension will be replaced with ".flac", or appended with ".flac" if the input file has no extension, and for decoding, the extension will be ".wav" for WAVE output and ".raw" for raw output). The original file is not deleted unless --delete-input-file is specified.

If you are encoding/decoding from stdin to a file, you should use the -o option like so:

- flac [options] -o outputfile
- flac -d [options] -o outputfile

which are better than:

- flac [options] > outputfile
- flac -d [options] > outputfile

since the former allows flac to seek backwards to write the STREAMINFO or RIFF WAVE header contents when necessary.

Also, you can force output data to go to stdout using -c.

To encode or decode files that start with a dash, use -- to signal the end of options, to keep the filenames themselves from being treated as options:

- flac -V -- -01-filename.wav

The encoding options affect the compression ratio and encoding speed. The format options are used to tell flac the arrangement of samples if the input file (or output file when decoding) is a raw file. If it is a RIFF WAVE, Wave64, RF64, or AIFF file the format options are not needed since they are read from the file's header.

In test mode, flac acts just like in decode mode, except no output file is written. Both decode and test modes detect errors in the stream, but they also detect when the MD5 signature of the decoded audio does not match the stored MD5 signature, even when the bitstream is valid.

flac can also re-encode FLAC files. In other words, you can specify a FLAC or Ogg FLAC file as an input to the encoder and it will decoder it and re-encode it according to the options you specify. It will also preserve all the metadata unless you override it with other options (e.g. specifying new tags, seekpoints, cuesheet, padding, etc.).

flac has been tuned so that the default settings yield a good speed vs. compression tradeoff for many kinds of input. However, if you are looking to maximize the compression rate or speed, or want to use the full power of FLAC's metadata system, see About the FLAC Format.

Tutorial

Some common **encoding** tasks using flac:

`flac abc.wav`
Encode `abc.wav` to `abc.flac` using the default compression setting. `abc.wav` is not deleted.

`flac --delete-input-file abc.wav`
Like above, except `abc.wav` is deleted if there were no errors.

`flac --delete-input-file -w abc.wav`
Like above, except `abc.wav` is deleted if there were no errors or warnings.

`flac --best abc.wav`
Encode `abc.wav` to `abc.flac` using the highest compression setting.

`flac --verify abc.wav`
Encode `abc.wav` to `abc.flac` and internally decode `abc.flac` to make sure it matches `abc.wav`.

`flac -o my.flac abc.wav`
Encode `abc.wav` to `my.flac`.

```
flac -T "TITLE=Bohemian Rhapsody" -T "ARTIST=Queen" abc.wav
```

Encode `abc.wav` and add some tags at the same time to `abc.flac`.

```
flac *.wav
```

Encode all .wav files in the current directory. NOTE: Wildcards on Windows

```
flac abc.aiff
```

Encode `abc.aiff` to `abc.flac`.

```
flac abc.rf64
```

Encode `abc.rf64` to `abc.flac`.

```
flac abc.w64
```

Encode `abc.w64` to `abc.flac`.

```
flac abc.flac --force
```

This one's a little tricky: notice that flac is in encode mode by default (you have to specify -d to decode) so this command actually recompresses `abc.flac` back to `abc.flac`. --force is needed to make sure you really want to overwrite `abc.flac` with a new version. Why would you want to do this? It allows you to recompress an existing FLAC file with (usually) higher compression options or a newer version of FLAC and preserve all the metadata like tags too.

Some common **decoding** tasks using flac:

```
flac -d abc.flac
```

Decode `abc.flac` to `abc.wav`. `abc.flac` is not deleted. NOTE: Without -d it means re-encode `abc.flac` to `abc.flac` (see above).

```
flac -d --force-aiff-format abc.flac
flac -d -o abc.aiff abc.flac
```

Two different ways of decoding `abc.flac` to `abc.aiff` (AIFF format). `abc.flac` is not deleted.

```
flac -d --force-rf64-format abc.flac
flac -d -o abc.rf64 abc.flac
```

Two different ways of decoding `abc.flac` to `abc.rf64` (RF64 format). `abc.flac` is not deleted.

```
flac -d --force-wave64-format abc.flac
flac -d -o abc.w64 abc.flac
```

Two different ways of decoding `abc.flac` to `abc.w64` (Wave64 format). `abc.flac` is not deleted.

```
flac -d -F abc.flac
```

Decode `abc.flac` to `abc.wav` and don't abort if errors are found (useful for recovering as much as possible from corrupted files).

flac has many other useful options, described below.

General Options	
-v, --version	Show the flac version number.
-h, --help	Show basic usage and a list of all options. Running flac without arguments shows the short help screen by default.
-H, --explain	Show detailed explanation of usage and all options. Running flac without arguments shows the short help screen by default.
-d, --decode	Decode (flac encodes by default). flac will exit with an exit code of 1 (and print a message, even in silent mode) if there were any errors during decoding, including

	when the MD5 checksum does not match the decoded output. Otherwise the exit code will be 0.
-t, --test	Test (same as -d except no decoded file is written). The exit codes are the same as in decode mode.
-a, --analyze	Analyze (same as -d except an analysis file is written). The exit codes are the same as in decode mode. This option is mainly for developers; the output will be a text file that has data about each frame and subframe.
-c, --stdout	Write output to stdout.
-s, --silent	Silent: do not show encoding/decoding statistics.
--totally-silent	Do not print anything of any kind, including warnings or errors. The exit code will be the only way to determine successful completion.
--no-utf8-convert	Do not convert tags from local charset to UTF-8. This is useful for scripts, and setting tags in situations where the locale is wrong. This option must appear before any tag options!
-w, --warnings-as-errors	Treat all warnings as errors (which cause flac to terminate with a non-zero exit code).
-f, --force	Force overwriting of output files. By default, flac warns that the output file already exists and continues to the next file.
-o filename, --output-name=filename	Force the output file name (usually flac just changes the extension). May only be used when encoding a single file. May not be used in conjunction with --output-prefix.
--output-prefix=string	Prefix each output file name with the given string. This can be useful for encoding/decoding files to a different directory. Make sure if your string is a path name that it ends with a trailing '/' slash.
--delete-input-file	Automatically delete the input file after a successful encode or decode. If there was an error (including a verify error) the input file is left intact.
--preserve-modtime	Output files have their timestamps/permissions set to match those of their inputs (this is default). Use --no-preserve-modtime to make output files have the current time and default permissions.
--keep-foreign-metadata	If encoding, save WAVE, Wave64, RF64, or AIFF non-audio chunks in FLAC metadata. If decoding, restore any saved non-audio chunks from FLAC metadata when writing the decoded file. Foreign metadata cannot be transcoded, e.g. WAVE chunks saved in a FLAC file cannot be restored when decoding to AIFF. Input and output must be regular files (not stdin or stdout).
--skip={#|mm:ss.ss} --skip={#|mm:ss,ss}	Skip over the first # of samples of the input. This works for both encoding and decoding, but not testing. The alternative form mm:ss.ss can be used to specify minutes, seconds, and fractions of a second. Note that the use of either a dot or a comma depends on the locale used for the system. Examples: --skip=123 : skip the first 123 samples of the input --skip=1:23.45 : skip the first 1 minute and 23.45 seconds of the input, with a locale using the point as decimal separator --skip=1:23,45 : skip the first 1 minute and 23.45 seconds of the input, with a locale using the comma as decimal separator
--until={#|[+|-]mm:ss.ss}	Stop at the given sample number for each input file. This works for both encoding

--until={#|[+|-]mm:ss,ss}	and decoding, but not testing. The given sample number is not included in the decoded output. The alternative form mm:ss.ss can be used to specify minutes, seconds, and fractions of a second. If a + sign is at the beginning, the --until point is relative to the --skip point. If a - sign is at the beginning, the --until point is relative to end of the audio. Note that the use of either a dot or a comma depends on the locale used for the system. Examples: --until=123 : decode only the first 123 samples of the input (samples 0-122, stopping at 123) --until=1:23.45 : decode only the first 1 minute and 23.45 seconds of the input --until=1:23,45 : decode only the first 1 minute and 23.45 seconds of the input, if your locale setting uses a comma as decimal separator --skip=1:00 --until=+1:23.45 : decode 1:00.00 to 2:23.45 --until=-1:23.45 : decode everything except the last 1 minute and 23.45 seconds --until=-0:00 : decode until the end of the input (the same as not specifying --until)
--ogg	When encoding, generate Ogg FLAC output instead of native FLAC. Ogg FLAC streams are FLAC streams wrapped in an Ogg transport layer. The resulting file should have an '.oga' extension and will still be decodable by flac. When decoding, force the input to be treated as Ogg FLAC. This is useful when piping input from stdin or when the filename does not end in '.oga' or '.ogg'. **NOTE:** Ogg FLAC files created prior to flac 1.1.1 used an ad-hoc mapping and do not support seeking. They should be decoded and re-encoded with flac 1.1.1 or later.
--serial-number=#	When used with --ogg, specifies the serial number to use for the first Ogg FLAC stream, which is then incremented for each additional stream. When encoding and no serial number is given, flac uses a random number for the first stream, then increments it for each additional stream. When decoding and no number is given, flac uses the serial number of the first page.

Analysis Options	
--residual-text	Includes the residual signal in the analysis file. This will make the file **very** big, much larger than even the decoded file.
--residual-gnuplot	Generates a gnuplot file for every subframe; each file will contain the residual distribution of the subframe. This will create a **lot** of files.

Decoding Options	
--cue=[#.#][-[#.#]]	Set the beginning and ending cuepoints to decode. The optional first #.# is the track and index point at which decoding will start; the default is the beginning of the stream. The optional second #.# is the track and index point at which decoding will end; the default is the end of the stream. If the cuepoint does not exist, the closest one before it (for the start point) or after it (for the end point) will be used. If those don't exist, the start of the stream (for the start point) or end of the stream (for

	the end point) will be used. The cuepoints are merely translated into sample numbers then used as --skip and --until. Examples: --cue=- : decode the entire stream --cue=4.1 : decode from track 4, index 1 to the end of the stream --cue=4.1- : decode from track 4, index 1 to the end of the stream --cue=-4.1 : decode from the beginning of the stream up to, but not including, track 4, index 1 --cue=2.1-2.4 : decode from track 2, index 1, up to, but not including, track 2, index 4 --cue=9.1-10.1 : decode from track 9 the way it would be played on a CD player; this works even if the CD has no 10th track.
-F, --decode-through-errors	By default flac stops decoding with an error and removes the partially decoded file if it encounters a bitstream error. With -F, errors are still printed but flac will continue decoding to completion. Note that errors may cause the decoded audio to be missing some samples or have silent sections.
--apply-replaygain-which-is-not-lossless[=<specification>]	Applies ReplayGain values while decoding. **WARNING: THIS IS NOT LOSSLESS. DECODED AUDIO WILL NOT BE IDENTICAL TO THE ORIGINAL WITH THIS OPTION**. The equals sign and <specification> is optional. If omitted, the default is 0aLn1. The <specification> is a shorthand notation for describing how to apply ReplayGain. All components are optional but order is important. '[]' means 'optional'. '|' means 'or'. '{}' means required. The format is: [<preamp>][a|t][l|L][n{0|1|2|3}] • <preamp> A floating point number in dB. This is added to the existing gain value. • a|t Specify 'a' to use the album gain, or 't' to use the track gain. If tags for the preferred kind (album/track) do not exist but tags for the other (track/album) do, those will be used instead. • l|L Specify 'l' to peak-limit the output, so that

	the ReplayGain peak value is full-scale. Specify 'L' to use a 6dB hard limiter that kicks in when the signal approaches full-scale. • n{0|1|2|3} Specify the amount of noise shaping. ReplayGain synthesis happens in floating point; the result is dithered before converting back to integer. This quantization adds noise. Noise shaping tries to move the noise where you won't hear it as much. 0 means no noise shaping, 1 means 'low', 2 means 'medium', 3 means 'high'. For example, the default of 0aLn1 means 0dB preamp, use album gain, 6dB hard limit, low noise shaping. --apply-replaygain-which-is-not-lossless=3 means 3dB preamp, use album gain, no limiting, no noise shaping. flac uses the ReplayGain tags for the calculation. If a stream does not have the required tags or they can't be parsed, decoding will continue with a warning, and no ReplayGain is applied to that stream.

Encoding Options

-V, --verify	Verify the encoding process. With this option, flac will create a parallel decoder that decodes the output of the encoder and compares the result against the original. It will abort immediately with an error if a mismatch occurs. -V increases the total encoding time but is guaranteed to catch any unforseen bug in the encoding process.
--lax	Allow encoder to generate non-Subset files. The resulting FLAC file may not be streamable or might have trouble being played in all players (especially hardware devices), so you should only use this option in combination with custom encoding options meant for archival.
--replay-gain	Calculate ReplayGain values and store them as FLAC tags, similar to VorbisGain. Title gains/peaks will be computed for each input file, and an album gain/peak will be computed for all files. All input files must have the same resolution, sample rate, and number of channels. Only mono and stereo files are allowed, and the sample rate must be one of 8, 11.025, 12, 16, 22.05, 24, 32, 44.1, or 48 kHz. Also note that this option may leave a few extra bytes in a PADDING block as the exact size of the tags is not known until all files are processed. Note that this option cannot be used when encoding to standard output (stdout).

--cuesheet=FILENAME

Import the given cuesheet file and store it in a CUESHEET metadata block. This option may only be used when encoding a single file. A seekpoint will be added for each index point in the cuesheet to the SEEKTABLE unless --no-cued-seekpoints is specified.

The cuesheet file must be of the sort written by CDRwin, CDRcue, EAC, etc. See also cuesheet syntax.

--picture={FILENAME|SPECIFICATION}

Import a picture and store it in a PICTURE metadata block. More than one --picture command can be specified. Either a filename for the picture file or a more complete specification form can be used. The SPECIFICATION is a string whose parts are separated by | (pipe) characters. Some parts may be left empty to invoke default values. FILENAME is just shorthand for ||||FILENAME. The format of SPECIFICATION is

```
  [TYPE]|[MIME-TYPE]|[DESCRIPTION]|
[WIDTHxHEIGHTxDEPTH[/COLORS]]|FILE
```

TYPE is optional; it is a number from one of:

- 0: Other
- 1: 32x32 pixels 'file icon' (PNG only)
- 2: Other file icon
- 3: Cover (front)
- 4: Cover (back)
- 5: Leaflet page
- 6: Media (e.g. label side of CD)
- 7: Lead artist/lead performer/soloist
- 8: Artist/performer
- 9: Conductor
- 10: Band/Orchestra
- 11: Composer
- 12: Lyricist/text writer
- 13: Recording Location
- 14: During recording
- 15: During performance
- 16: Movie/video screen capture
- 17: A bright coloured fish
- 18: Illustration
- 19: Band/artist logotype
- 20: Publisher/Studio logotype

The default is 3 (front cover). There may only be one picture each of type 1 and 2 in a file.

MIME-TYPE is optional; if left blank, it will be detected from the file. For best compatibility with players, use pictures with MIME type `image/jpeg` or `image/png`. The MIME type can also be --> to mean that FILE is actually a URL to an image, though this use is discouraged.

DESCRIPTION is optional; the default is an empty string.

<table>
<tr><td></td><td>The next part specfies the resolution and color information. If the MIME-TYPE is <code>image/jpeg</code>, <code>image/png</code>, or <code>image/gif</code>, you can usually leave this empty and they can be detected from the file. Otherwise, you must specify the width in pixels, height in pixels, and color depth in bits-per-pixel. If the image has indexed colors you should also specify the number of colors used. When manually specified, it is not checked against the file for accuracy.

FILE is the path to the picture file to be imported, or the URL if MIME type is -->

For example, the specification |image/jpeg|||../cover.jpg will embed the JPEG file at <code>../cover.jpg</code>, defaulting to type 3 (front cover) and an empty description. The resolution and color info will be retrieved from the file itself.

The specification 4|-->|CD|320x300x24/173|http://blah.blah/backcover.tiff will embed the given URL, with type 4 (back cover), description "CD", and a manually specified resolution of 320x300, 24 bits-per-pixel, and 173 colors. The file at the URL will not be fetched; the URL itself is stored in the PICTURE metadata block.</td></tr>
<tr><td>--sector-align</td><td>Align encoding of multiple CD format files on sector boundaries. This option is only allowed when encoding files all of which have a 44.1kHz sample rate and 2 channels. With --sector-align, the encoder will align the resulting .flac streams so that their lengths are even multiples of a CD sector (1/75th of a second, or 588 samples). It does this by carrying over any partial sector at the end of each file to the next stream. The last stream will be padded to alignment with zeroes.

This option will have no effect if the files are already aligned (as is the normally the case with WAVE files ripped from a CD). flac can only align a set of files given in one invocation of flac.

<b>WARNING:</b> The ordering of files is important! If you give a command like 'flac --sector-align *.wav' the shell may not expand the wildcard to the order you expect. To be safe you should 'echo *.wav' first to confirm the order, or be explicit like 'flac --sector-align 8.wav 9.wav 10.wav'.

<b>NOTE:</b>This option is DEPRECATED and may not exist in future version of flac. shntool provides similar functionality.</td></tr>
<tr><td>--ignore-chunk-sizes</td><td>When encoding to flac, ignore the file size headers in WAV and AIFF files to attempt to work around problems with over-sized or malformed files.

WAV and AIFF files both have an unsigned 32 bit numbers in the file header which specifes the length of audio data. Since this number is unsigned 32 bits, that limits the size of a valid file to being just over 4 Gigabytes. Files larger than this are mal-formed, but should be read correctly using this option.</td></tr>
</table>

-S {#|X|#x|#s}, --seekpoint={#|X|#x|#s}	Include a point or points in a SEEKTABLE: • # : a specific sample number for a seek point • X : a placeholder point (always goes at the end of the SEEKTABLE) • #x : # evenly spaced seekpoints, the first being at sample 0 • #s : a seekpoint every # seconds; # does not have to be a whole number, it can be, for example, 9.5, meaning a seekpoint every 9.5 seconds You may use many -S options; the resulting SEEKTABLE will be the unique-ified union of all such values. With no -S options, flac defaults to '-S 10s'. Use --no-seektable for no SEEKTABLE. **NOTE:** -S #x and -S #s will not work if the encoder can't determine the input size before starting. **NOTE:** if you use -S # and # is >= samples in the input, there will be either no seek point entered (if the input size is determinable before encoding starts) or a placeholder point (if input size is not determinable).
-P #, --padding=#	Tell the encoder to write a PADDING metadata block of the given length (in bytes) after the STREAMINFO block. This is useful if you plan to tag the file later with an APPLICATION block; instead of having to rewrite the entire file later just to insert your block, you can write directly over the PADDING block. Note that the total length of the PADDING block will be 4 bytes longer than the length given because of the 4 metadata block header bytes. You can force no PADDING block at all to be written with --no-padding. The encoder writes a PADDING block of 8192 bytes by default (or 65536 bytes if the input audio stream is more than 20 minutes long).
-T FIELD=VALUE, --tag=FIELD=VALUE	Add a FLAC tag. The comment must adhere to the Vorbis comment spec (which FLAC tags implement), i.e. the FIELD must contain only legal characters, terminated by an 'equals' sign. Make sure to quote the comment if necessary. This option may appear more than once to add several comments. NOTE: all tags will be added to all encoded files.
--tag-from-file=FIELD=FILENAME	Like --tag, except FILENAME is a file whose contents will be read verbatim to set the tag value. The contents will be converted to UTF-8 from the local charset. This can be used to store a cuesheet in a tag (e.g. --tag-from-file="CUESHEET=image.cue"). Do not try to store binary data in tag fields! Use APPLICATION blocks for that.
-b #, --blocksize=#	Specify the block size in samples. Subset streams must use one of 192/576/1152/2304/4608/256/512/1024/2048/4096 (and 8192/16384 if the sample rate is >48kHz). The reference encoder uses the same block size for the entire stream.
-m, --mid-side	Enable mid-side coding (only for stereo streams). Tends to increase compression by a few percent on average. For each block both the stereo pair and mid-side versions of the block will be encoded, and smallest resulting frame will be stored.
-M, --adaptive-mid-side	Enable adaptive mid-side coding (only for stereo streams). Like -

	m but the encoder adaptively switches between independent and mid-side coding, which is faster but yields less compression than -m (which does an exhaustive search).
-0 .. -8	Fastest compression .. highest compression. The default is -5.
-0, --compression-level-0	Synonymous with -l 0 -b 1152 -r 3
-1, --compression-level-1	Synonymous with -l 0 -b 1152 -M -r 3
-2, --compression-level-2	Synonymous with -l 0 -b 1152 -m -r 3
-3, --compression-level-3	Synonymous with -l 6 -b 4096 -r 4
-4, --compression-level-4	Synonymous with -l 8 -b 4096 -M -r 4
-5, --compression-level-5	Synonymous with -l 8 -b 4096 -m -r 5
-6, --compression-level-6	Synonymous with -l 8 -b 4096 -m -r 6 -A tukey(0.5);partial_tukey(2)
-7, --compression-level-7	Synonymous with -l 12 -b 4096 -m -r 6 -A tukey(0.5);partial_tukey(2)
-8, --compression-level-8	Synonymous with -l 12 -b 4096 -m -r 6 -A tukey(0.5);partial_tukey(2);punchout_tukey(3)
--fast	Fastest compression. Currently synonymous with -0
--best	Highest compression. Currently synonymous with -8
-e, --exhaustive-model-search	Exhaustive model search (expensive!). Normally the encoder estimates the best model to use and encodes once based on the estimate. With an exhaustive model search, the encoder will generate subframes for every order and use the smallest. If the max LPC order is high this can significantly increase the encode time but can shave off another 0.5%.
-A "function", --apodization="function"	Window audio data with given the apodization function. The functions are: bartlett, bartlett_hann, blackman, blackman_harris_4term_92db, connes, flattop, gauss(STDDEV), hamming, hann, kaiser_bessel, nuttall, rectangle, triangle, tukey(P), partial_tukey(n[/ov[/P]]), punchout_tukey(n[/ov[/P]]), welch. For gauss(STDDEV), STDDEV is the standard deviation (0<STDDEV<=0.5). For tukey(P), P specifies the fraction of the window that is tapered (0<=P<=1; P=0 corresponds to "rectangle" and P=1 corresponds to "hann"). For partial_tukey(n) and punchout_tukey(n), n apodization functions are added that span different parts of each block. Values of 2 to 6 seem to yield sane results. If necessary, an overlap can be specified, as can be the taper parameter, for example partial_tukey(2/0.2) or partial_tukey(2/0.2/0.5). ov should be smaller than 1 and can be negative. Please note that P, STDDEV and ov are locale specific, so a comma as decimal separator might be required instead of a dot. More than one -A option (up to 32) may be used. Any function that is specified erroneously is silently dropped. The encoder chooses suitable defaults in the absence of any -A options; any -A option specified replaces the default(s). When more than one function is specified, then for every subframe the encoder will try each of them separately and choose

	the window that results in the smallest compressed subframe. Multiple functions can greatly increase the encoding time.
-l #, --max-lpc-order=#	Specifies the maximum LPC order. This number must be <= 32. For Subset streams, it must be <=12 if the sample rate is <=48kHz. If 0, the encoder will not attempt generic linear prediction, and use only fixed predictors. Using fixed predictors is faster but usually results in files being 5-10% larger.
-q #, --qlp-coeff-precision=#	Specifies the precision of the quantized LP coefficients, in bits. The default is -q 0, which means let the encoder decide based on the signal. Unless you really know your input file it's best to leave this up to the encoder.
-p, --qlp-coeff-precision-search	Do exhaustive LP coefficient quantization optimization. This option overrides any -q option. It is expensive and typically will only improve the compression a tiny fraction of a percent. -q has no effect when -l 0 is used.
-r [#,]#, --rice-partition-order=[#,]#	Set the [min,]max residual partition order. The min value defaults to 0 if unspecified. By default the encoder uses a single Rice parameter for the subframe's entire residual. With this option, the residual is iteratively partitioned into 2^min# .. 2^max# pieces, each with its own Rice parameter. Higher values of max# yield diminishing returns. The most bang for the buck is usually with -r 2,2 (more for higher block sizes). This usually shaves off about 1.5%. The technique tends to peak out about when blocksize/(2^n)=128. Use -r 0,15 to force the highest degree of optimization.

Format Options	
--endian={big|little}	Specify big-endian or little-endian byte order in the raw file.
--channels=#	Specify the number of channels in the raw file.
--bps=#	Specify the number of bits per sample in the raw file.
--sample-rate=#	Specify the sample rate of the raw file.
--sign={signed|unsigned}	Specify that the samples in the raw file are signed or unsigned (the default is signed).
--input-size=#	Specify the size of the raw input in bytes. If you are encoding raw samples from stdin, you must set this option in order to be able to use --skip, --until, --cuesheet, or other options that need to know the size of the input beforehand. If the size given is greater than what is found in the input stream, the encoder will complain about an unexpected end-of-file. If the size given is less, samples will be truncated.
--force-raw-format	Treat the input file (or output file if decoding) as a raw file, regardless of the extension.
--force-aiff-format	Force the decoder to output AIFF format. This option is not needed if the output filename (as set by -o) ends with .aif or .aiff. Also, this option has no effect when encoding since input AIFF is auto-detected.
--force-rf64-format	Force the decoder to output RF64 format. This option is not needed if the output filename (as set by -o) ends with .rf64. Also, this option has no effect when encoding since input RF64 is auto-detected.
--force-wave64-format	Force the decoder to output Wave64 format. This option is not needed if the output filename (as set by -o) ends with .w64. Also, this option has no effect when

	encoding since input Wave64 is auto-detected.

Negative Options	
--no-adaptive-mid-side --no-cued-seekpoints --no-decode-through-errors --no-delete-input-file --no-escape-coding --no-exhaustive-model-search --no-ignore-chunk-sizes --no-lax --no-mid-side --no-ogg --no-padding --no-preserve-modtime --no-qlp-coeff-prec-search --no-residual-gnuplot --no-residual-text --no-sector-align --no-seektable --no-silent --no-verify --no-warnings-as-errors	Can all be used to turn off a particular option.

Option Index