Brother John

BeSweet Commandline Reference

Version 2012-01-11
http://brother-john.net/besweet-reference.html

Table of Contents

Complete Syntax Overview ..... **III**

1 Basics **1**
1.1 Installation 1
1.2 BeSweet's Syntax 3

2 Global BeSweet Configuration **5**
2.1 The Section `-core()` 5
2.1.1 Basic Options 5
2.1.2 Options for VOB Input 7
2.1.3 Destination Format Switches 7
2.1.4 Tweaking Switches 9
2.2 The Section `-split()` 10
2.3 The Section `-plugin()` 11

3 Decoders and Audio Manipulation **12**
3.1 The Section `-ota()` 12
3.1.1 Normalizing Options 12
3.1.2 Delay Options 13
3.1.3 Options for Sampling Frequency and Framerate 14
3.2 The Section `-soundtouch()` 15
3.3 The Sections `-shibatch()` and `-ssrc()` 16
3.4 The Section `-boost()` 17
3.5 The Section `-azid()` 18
3.5.1 Normalization and Dynamic Compression 18
3.5.2 Downmix Options 19
3.5.3 Options for Controlling Individual Channels 21

4 Encoders **23**
4.1 The Section `-ogg()` 23
4.2 The Section `-lame()` 24
4.2.1 LAME up to Version 3.96 24
4.2.2 LAME Version 3.97 and Newer 25
4.2.3 Special Options 27
4.3 The Sections `-mp2enc()` and `-toolame()` 28
4.4 The Section `-ac3enc()` 29

4.5 The Section `-bsn()` for Aften AC3 29
4.6 The Section `-bsn()` for Flake FLAC 30
4.7 The Section `-bsn()` for Nero AAC 31
4.8 The Section `-dimzon()` for Winamp AAC 32

Complete Syntax Overview

All elements are clickable links to their full explanation.

```
BeSweet.exe -core() [-split()] [-plugin()] [-ota()]
[-soundtouch()] [-shibatch()|-ssrc()] [-boost()] [-azid()]
[-ogg()|-bsn()|-lame()|-mp2enc()|-toolame()|-ac3enc()|-dimzon() ]

-core( -input <source file> -output <destination file>
[-logfile <log file>|-logfilea <log file>]
[-seek|-substream <stream number>]
[-2ch|-6ch|-6CH|-6chfloat|-6chwav|-5chaiff|-6chaiff|-6chliff|
-6chogg|-wavmp3|-ddwav|-payload|-6chnothing|-2chnothing]
[-noid3] [-be] )

-split( [-start <sec>] [-end <sec>] )

-plugin( -name <DLL name> -func <name> [-6ch] )

-ota( [-norm <level>|-g {<dB>|max|peak}|-G <level>|-hybridgain]
[-d <ms>] [-e <ms>] [-sc <resolution>] [-fs <Hz>]
[-r <source fps> <destination fps>] )

-soundtouch( [-r <source fps> <dest. fps>|-tempo <percent>|
-pitch <semitones>|-rate <percent>] [-quick] [-naa] )

-shibatch( [--rate <Hz>] [--normalize] [--att <dB>] )
-ssrc( [--rate <Hz>] [--normalize] [--att <dB>] )

-boost( {/b=<level>|/b2=<level>|/b3=<level>} [/l=<value>] )

-azid( [-g {max|<level>}|--maximize] [-c {none|light|normal|
heavy|inverse}] [-d <front>/<rear>] [-s {stereo|dpl|surround|
dplii|surround2}] [-S {<level>|BSI}] [-C {<level>|BSI}]
[-L <level>] [-l <level>] [-f {0|1}] [-n {0|1}]
[-o <channel order>] [--ch{<number>|<name>={light|normal|
heavy|inverse},<dB>] )
```

```
-ogg( [-q <quality>|{-b <kbps> [-m <kbps>] [-M <kbps>]}] [-6ch] )

-lame( <encoding mode> [<more options>] )
-lame( --preset {{[fast] medium|[fast] standard|[fast] extreme|
insane}|cbr <kbps>|<kbps>} [<more options>] )
-lame( -v --vbr-{old|new} -V <quality> [-b <kbps>] [-B <kbps>]
[<more options>] )
-lame( {-b <kbps>|--abr <kbps> [-b <kbps>] [-B <kbps>]}
[<more options>] )
-lame( <encoding mode> [--resample <Hz>] [--lowpass <kHz>]
[-m {s|j|f|m}] [-p] [-Y] )

-mp2enc( [-b <kbps>|-v <quality>] [-m {s|j|d|m}] [-e] )
-toolame( [-b <kbps>|-v <quality>] [-m {s|j|d|m}] [-e] )

-ac3enc( [-b <kbps>] [-6ch] )

-bsn( -exe aften.exe {-b <kbps>|-q <quality>} [-6chnew] )

-bsn( -exe flake.exe [-level <level>] [-6chnew] )

-bsn( [-vbr <quality>|-abr <kbps>|-cbr <kbps>] [-6chnew]
[-aacprofile_{lc|he|hev2}] [-hintforstreaming] )

-dimzon( -dllname WA_aac.dll -cbr <kbps> [-lc] [-high] [-pns]
[-mpeg2] [-mode {1|2|3|4|5}] [-6chnew] )
```

1 Basics

This document contains an overview of all of BeSweet's commandline options and their interaction. The Reference is based on DSPguru's BeSweet version 1.5 beta 31 as well as bsn.dll und WA_aac.dll by Kurtnoise13. If you have questions a good place to start is the BeSweet FAQ on doom9.org (Warning, partly outdated!): http://forum.doom9.org/showthread.php?t=15738

At the moment the Reference is missing the command lines for 3GP AAC, FAAC and Speex.

1.1 Installation

As easy as it might seem at first, there are a couple of traps in BeSweet's setup you might fall into. This chapter (hopefully) will steer you around them. For a complete BeSweet installation we need the following components:

► BeSweet
BeSweet v1.5b31 from BeSweet's homepage in section "beta".
http://besweet.notrace.dk/

► VOBInput.dll
In case you want to transcode audio directly from VOB files.
http://besweet.notrace.dk/

► Lame_enc.dll
For encoding MP3s. Best download site for LAME is RareWares.org. I prefer the current stable version.
http://www.rarewares.org/mp3-lame-bundle.php

► Libvorbis.dll
For encoding Ogg Vorbis. RareWares.org provides the latest package of *libvorbis.dll using aoTuVb*.

http://www.rarewares.org/ogg-libraries.php

► 2Lame
MP2 encoding with 2Lame needs *toolame.dll* not included in the BeSweet package. BeSweet only contains the mp2enc MP2 encoder.
http://sourceforge.net/projects/toolame/

► Nero
To encode AAC with Nero you do not need the full Nero package any more. BeSweet uses the free Nero commandline encoder.
http://www.nero.com/eng/technologies-aac-codec.html

► Winamp
For AAC encoding with Winamp you first need *WA_aac.dll* available in the BeLight download directory. Then you need a Winamp version that comes with an AAC encoder.
http://kurtnoise.free.fr/BeLight/
http://www.winamp.com/

► Aften
For encoding to AC-3. Aften's SourceForge project only provides the source code. Executables are available on a different SourceForge project.
http://win32builds.sourceforge.net/aften/

► Flake
For encoding to FLAC. Source code and binaries are available on SourceForge.
http://sourceforge.net/projects/flake-enc/

► Bsn.dll
Bsn.dll is required for Nero, Winamp, Aften and Flake. The BeLight download directory provides the latest version.
http://kurtnoise.free.fr/BeLight/

Once you have all necessary components, first unzip the BeSweet archive to some folder, e. g. *C:\Program Files\BeSweet.* Then install Winamp, if want to use that encoder.

Then all of the following files from the downloaded archives go into the BeSweet folder.

► **VOBInput.dll** and **toolame.dll**,
► **bsn.dll** from the bsn archive,
► **lame_enc.dll** from the LAME archive,
► **libvorbis.dll** from the Vorbis archive,
► **aften.exe** from the Aften archive, preferably the one with matching optimization for you CPU,
► **flake.exe** from the Flake archive, preferably the one with matching optimization for you CPU,
► **neroAacEnc.exe** from the Nero archive.

Winamp AAC encoding also needs these two files:

► **WA_aac.dll** from the downloaded archive,
► **enc_aacplus.dll** from the Winamp install's *Plugins* subfolder.

Now BeSweet is fully functional.

1.2 BeSweet's Syntax

Syntax

```
BeSweet.exe -core() [-split()] [-plugin()] [-ota()]
[-soundtouch()] [-shibatch()|-ssrc()] [-boost()] [-azid()]
[-ogg()|-lame()|-mp2enc()|-toolame()|-ac3enc()|-bsn()|-dimzon()]
```

Sections

Only the `-core()` section is mandatory. All the others may be omitted or combined in almost any way, except for the encoder sections. Only one of those may be used. Here is the shortest possible BeSweet command line:

```
BeSweet.exe -core( -input <source> -output <destination> )
```

BeSweet will create a 128 kbit/s CBR stereo MP3.

Spaces

Sections are separated with a space each. Additionally a space must be there after every opening bracket and before every closing bracket. Options inside a section must be separated by a space, too. An example (dots • mark spaces):

```
-core(•-input•source.ac3•-output•dest.wav•-2ch•)•-ota(•-d•80•)
```

Decimal separator
Regardless of system settings BeSweet always uses a dot as decimal separator, never a comma. I.e. ½ always must be written as `0.5`, never as `0,5`.

Default values
Some options have a default value (common with `-azid()`). BeSweet uses the default value, if such an option is omitted. If you want to, for example, tell Azid to perform a Dolby Pro Logic downmix, you don't have to specify the appropriate option (`-s surround`) explicitly because BeSweet automatically assumes `-s surround`, if `-s` is not present. All functions without a default value are only active if given explicitly. If you omit OTA's delay option (`-d`) , no delay will be applied to the output file.

Case sensitive options
Options of most sections are case sensitive. E.g. `-6ch` and `-6CH` (`-core()` section) are two different options. That's why you should always use options exactly as given in this reference.

2 Global BeSweet Configuration

2.1 The Section `-core()`

Syntax

```
-core( -input -output [-logfile|-logfilea] [-seek|-substream] [-
2ch|-6ch|-6CH|-6chfloat|-6chwav|-5chaiff|-6chaiff|-6chliff|
-6chogg|-wavmp3|-ddwav|-payload|-6chnothing|-2chnothing]
[-noid3] [-be] )
```

This section contains BeSweet's basic configuration, including definitions of source, destination and log files as well as switches for output formats not needing an encoder section.

2.1.1 Basic Options

Options `-input` and `-output` are both required. All the others are optional.

`-input <source file>`

Path and name of source file. Of course, relative paths are possible. Paths containing spaces need to be enclosed in quotes according to Windows conventions. File names containing wildcards (* and ?) are not allowed.

```
-input "D:\Audio\Stream 1.ac3"
```

BeSweet accepts these types of source files:

► **MPEG audio (*.MP3, *.MP2, *.MPA).** MPEG-1 Layer II and Layer III in mono and stereo.

► **Dolby Digital (*.AC3).** Mono, stereo and multichannel.

► **Ogg Vorbis (*.OGG).** Mono, stereo and multichannel.

► **Wave (*.WAV).** Mono, stereo and multichannel waves. 16 bit waves only up to 2 GBytes. 32 bit floating point waves without a size limit.

► **Raw PCM stream (*.PCM).** "Headerless waves". BeSweet assumes 16 bit, stereo, 48 kHz. Different sampling rates may be given using `-ota( -fs <Hz> )`.

► **AVIs (*.AVI).** AVI files containing AC3, MPEG or PCM audio streams. PCM may be 8 bit or 16 bit with mono, stereo or multiple channels.

► **VOBs (*.VOB).** DVD video object files containing AC3 and/or PCM audio. PCM may be 8 bit or 16 bit with mono, stereo or multiple channels. 20 bit and 24 bit are not supported.
► **BeSweet list file (*.LST).** Plaintext file containing paths to a set of source files (one path per line). Those files will be encoded to one output file.
► BeSweet assumes 48 kHz sampling frequency. Different frequencies may be given using `-ota( -fs <Hz> )`.
► **BeSweet muxing file (*.MUX).** Plaintext file similar to the list file. A muxing file contains paths for up to six mono waves (one path per line), that will be processed like one multichannel wave. The order of the file names defines the output file's channel order.

-output <destination file>

Path and name of destination file. Of course, relative paths are possible. Paths containing spaces need to be enclosed in quotes according to Windows conventions. File names containing wildcards (* and ?) are not allowed.

```
-output "D:\Audio\Stream 1 transcoded.ogg"
```

Given file extension does *not* define the output format, which is handled by destination format switches or an encoder section. BeSweet does not correct wrong extensions. This means it is (not very useful but) perfectly possible to create for example an Ogg Vorbis file with extension .ac3.

Warning: BeSweet will overwrite existing output files will without asking for confirmation!

-logfile <log file>

Path and name of the log file, into which BeSweet writes information about the transcoding process. Without this option BeSweet will output to the screen.

```
-logfile "D:\Audio\Logfile.log"
```

-logfilea <log file>

Similar to `-logfile`, but appends new logs to an existing file instead of overwriting.

```
-logfilea "D:\Audio\Logfile.log"
```

2.1.2 Options for VOB Input

`-seek`

Displays stream numbers (see `-substream`) for every audio track contained inside a VOB. Does not transcode anything. This switch only requires a source file given using `-input`. Additional options will be ignored.

`-substream <stream number>`

The audio track with given stream number will be transcoded. Numbers are hexa-decimal values. A DVD normally contains AC3 inside streams `0x80` to `0x88`, PCM inside `0xA0` to `0xA7` and MPEG audio inside `0xC0` to `0xC7`. Streams `0x89` to `0x8F` are reserved for optional audio formats like DTS. Standalone players without a proper decodes either ignore those or just output the digital stream.

If input file is a VOB and `-substream` omitted, BeSweet automatically decodes the file's first stream.

```
-substream 0x83
```

2.1.3 Destination Format Switches

Apart from exceptions (`-6chogg`, `-wavmp3`, `-ddwav`) these switches are only needed when no encoder section is defined. In most of those cases BeSweet won't be used for transcoding, but for decoding or sound processing only.

`-2ch`

Creates a 16 bit stereo wave. Channel order is: FL, FR.

`-6ch`

Creates six 16 bit mono waves, one for every source channel. If the source lacks a channel, the corresponding wave will contain silence. Assumed channel order is: FL, FR, C, LFE, SL, SR.

-6CH

Creates six 16 bit mono waves, one for every source channel. If the source lacks a channel, the corresponding wave will contain silence. Assumed channel order is: FL, C, FR, SL, SR, LFE.

-6chfloat

Creates six 32 bit floating point mono waves, one for every source channel. If the source lacks a channel, the corresponding wave will contain silence. Assumed channel order is: FL, FR, C, LFE, SL, SR.

-6chwav

Creates a 16 bit multichannel wave. Channel order of this file is: FL, FR, C, LFE, SL, SR.

-5chaiff

Creates a 16 bit, 5 channel, big endian AIFF file. Channel order of this file is: FL, FR, C, SL, SR.

-6chaiff

Creates a 16 bit, 6-channel, big endian AIFF file. Channel order of this file is: FL, FR, C, LFE, SL, SR.

-6chliff

Creates a 16 bit, 6-channel, little endian AIFF file. Channel order of this file is: FL, FR, C, LFE, SL, SR.

-6chogg

Together with `-ogg()` creates a multichannel Ogg Vorbis file. If no `-ogg()` section is present, BeSweet encodes with VBR quality `0.800`. Assumed channel order of input file is: FL, C, FR, SL, SR, LFE. Only use this switch when the input file really has this unusual channel order.

-wavmp3

Together with `-lame()` transcodes to MP3 and writes an additional wave header to the file. That is useful for muxing MP3s with programs like VirtualDub (but not VirtualDubMod), that only can handle waves. Only use the switch for those special cases. Without a `-lame()` section an 128 kbit/s CBR stereo MP3 will be created.

-ddwav

Creates a Dolby Digital wave together with `-ac3enc()`. Essentially that's an AC3 with additional wave header, similar to `-wavmp3`. Without an `-ac3enc()` section BeSweet will encode using 640 kbit/s.

-payload

Does not decode anything, but copies the source audio unprocessed. Useful for demuxing tracks from VOBs, for splitting files without decoding, for inserting delays etc.

2.1.4 Tweaking Switches

-6chnothing

Decodes a 6-channel stream without writing data to disk. Used by AacMachine to normalise tracks from a VOB.

-2chnothing

Decodes a stereo stream without writing data to disk. Used by AacMachine to normalise tracks from a VOB.

-noid3

Disables the ID3V1 tag that is usually added to the output file.

-be

Forces BeSweet to assume a 16 bit big endian wave as source file.

2.2 The Section -split()

Syntax
```
-split( [-start] [-end] )
```

This section provides partial transcoding of a source file. You are not forced to use both -start and -end. Without -start starting point is the beginning of the file, without -end transcoding continues to the end of the file.

Both values are absolute. E.g. a starting point at 6 seconds and an end point at 10 seconds produces an output file of 4 seconds in length.

-start <sec>

Sets the starting point for processing the source file to the given value in seconds. Fractions are allowed. The example below sets the starting point to 10 seconds and 80 milliseconds.

```
-start 10.080
```

-end <sec>

Sets the end point for processing the source file to the given value in seconds. Fractions are allowed. The example below sets the end point to 90 seconds and 525 milliseconds.

```
-end 90.525
```

2.3 The Section -plugin()

Syntax
```
-plugin( -name -func [-6ch] )
```

Features can be added to BeSweet using plugins, whose functions are invoked using the -plugin() section.

-name <DLL name>

Sets the plugin's file name.
```
-name MyPlugin.dll
```

-func <name>

Executes a function called <name> provided by the plugin.

-6ch

Inputs 6-channel data to the plugin. Necessary, if the plugin performs downmixing or similar actions.

3 Decoders and Audio Manipulation

3.1 The Section `-ota()`

Syntax
`-ota( [-norm|-g|-G|-hybridgain] [-d] [-e] [-sc] [-fs] [-r] )`

OTA is an acronym for “overall track adjustment”. Mainly it is used for normalising audio streams and inserting delays.

3.1.1 Normalizing Options

`-norm <level>`

Normalises the file before encoding (pre gain normalisation) to a given percentage. Utilises a two-pass method. The first pass finds the audio stream’s peak level. Using this value and the given `<level>` the appropriate gain value is then calculated and applied in the second pass, where the stream gets encoded to its final format. `<Level>` is a floating point value between `0` and `1`, where `1` means 100% or maximum possible normalisation.
`-norm 0.97`

`-g {<dB>|max|peak}`

Used for raising or lowering audio volume by a fixed number of decibels before encoding (pre gain). E. g. `5db` raises every sample by 5 dB, `-5dB` attenuates by 5 dB.
`-g -5db`
`max` uses the highest possible value not distorting the sound, working identical to `-norm 1`.
`-g max`
`peak` uses the value stored within the source waves peak chunk. If no such value exists or the source file is not a wave, `peak` works identical to `max`.
`-g peak`
Compared to Azid’s `-g` option, OTA’s `-g` is a bit slower, but also a bit more accurate. Furthermore if a listfile is used, it normalises all sources as one, eliminating the problem of volume jumps in the output file.

-G <level>

Normalizes the output file after encoding (post gain normalisation) to a given percentage. The value is a floating point number between 0 and 1, where 1 means 100 % or maximum possible normalisation.

If the destination file is not MP2, MP3 or Vorbis, -G works identical to -norm. However this is not true for AAC, which always needs pre gain (-g or -norm) specified explicitly.

```
-G 0.97
```

-hybridgain

Combines a fixed pre gain value with post gain normalisation depending on source and destination formats. The following table is only valid for stereo output. If you encode to multichannel, pre gain is always 0 dB.

Source file	DNR [1]	Pre gain with dynamic compression [2]				
		none	light	normal	heavy	inverse
6-channel AC3	on	7 dB	8 dB	10 dB	14 dB	5 dB
Stereo AC3	off	0 dB	0 dB	0 dB	0 dB	0 dB

1) Dialog Normalization Reduction, Azid's -n 2) Azid's -c

After encoding BeSweet always post gain normalizes to 97 %. For MP3 lame's scale option will be set to 1, too.

If source is not AC3 or destination is not MP2, MP3 or Vorbis, -hybridgain works identical to -g max. However this is not true for AAC, which always needs pre gain (-g or -norm) specified explicitly.

3.1.2 Delay Options

-d <ms>

Inserts given number of milliseconds of silence at start of the file or removes first milliseconds (for negative values).

```
-d 112
-d -80
```

-e <ms>

Appends given number of milliseconds of silence to end of file.
`-e 1600`

-sc <resolution>

Sets minimum sample resolution for delay operations. Used by AacMachine to only insert complete AAC frames for delays.

3.1.3 Options for Sampling Frequency and Framerate

-fs <Hz>

Skips BeSweet's automatic recognition of sampling frequency, enforcing the given value in hertz.
`-fs 44100`
This function does not convert sampling frequencies! BeSweet only without further checks assumes a source file with given frequency. Useful for sources with unknown sampling frequency, e. g. raw PCM files.

-r <source fps> <destination fps>

Converts the audio stream to a different framerate changing pitch as well as track length. This function is useful for example for NTSC-to-PAL conversion, i. e. when converting the video stream's framerate.

Values are integers. BeSweet interprets the last three numbers as fractions. Accordingly the following example would convert from 23.976 fps to 25.000 fps.

`-r 23976 25000`
Conversion is not fully available for every kind of 6-channel output. Only possibilities are creating 5.1 AC3s and separate mono waves.

3.2 The Section -soundtouch()

Syntax
`-soundtouch( [-r|-tempo|-pitch|-rate] [-quick] [-naa] )`

This section provides features to adjust speed and pitch of an audio stream, but only for mono and stereo output. For multichannel output OTA's `-r` might be an alternative. Generally `-soundtouch()` output should be of higher quality than `-ota ( -r )`.

-r <source fps> <destination fps>

Converts the audio stream to a different framerate without changing pitch. This function is useful for example for NTSC-to-PAL conversion, i. e. when converting the video stream's framerate.

Values are integers. BeSweet interprets the last three numbers as fractions. Accordingly the following example would convert from 23.976 fps to 25.000 fps.

```
-r 23976 25000
```

-tempo <percent>

Changes audio speed by the given percentage without affecting pitch. Valid values lie between `-95` and `+5000`. Negative values are for slowing down, positive ones for speeding up.

```
-tempo -20
```

-pitch <semitones>

Changes pitch by given number of halftones without changing audio speed. Possible values are `-60` to `+60`. Negative values are for lowering, positive ones for raising pitch.

```
-pitch +15
```

-rate <percent>

Changes audio speed by given percentage, affecting pitch as well as track length. This option works similar to OTA's `-r`. Values may be given between `-95` and `+5000`, negative values slowing down and positive ones speeding up.

```
-rate -20
```

-quick

Uses a different algorithm for calculations, sacrificing quality for improved speed.

-naa

Disables anti-aliasing to gain speed at the expense of some quality.

3.3 The Sections -shibatch() and -ssrc()

Syntax

```
-shibatch( [--rate] [--normalize] [--att] )
-ssrc( [--rate] [--normalize] [--att] )
```

These sections convert the audio stream's sampling frequency. However, some problems are known. E.g. upsampling from 44.1 kHz to 48 kHz won't always work correctly. For such cases you should use an external program like SSRC.exe. Both section names, `-shibatch()` and `-ssrc()`, are aliases for the same functionality.

--rate <Hz>

Defines output sampling rate in hertz.

```
--rate 24000
```

--normalize

Normalises volume to 100 %. You should use OTA's or Azid's normalising instead.

--att <dB>

Attenuates output by given number of decibels.

```
--att 3
```

3.4 The Section -boost()

Syntax
`-boost( {/b|/b2|/b3} [/l] )`

This section is used for dynamic compression, just like Azid's `-c`. To avoid interference between the two function resulting in audio artefacts you should always use only one of them.

/b=<level>

Defines the dynamic compression's strength using an algorithm by LigH. Valid arguments range from `1` to `10`. With a value around `3` this option is especially useful for stereo sources.
`/b=3`

/b2=<level>

Defines the dynamic compression's strength using an algorithm by DSPguru. Valid arguments range from `1` to `10`. With a value around `4` this option is especially useful for 6-channel sources.
`/b2=4`

/b3=<level>

Defines the dynamic compression's strength using an algorithm by Tera. Valid arguments range from `1` to `10`.
`/b3=5`

/l=<value>

Default: `0.99`

Defines maximum allowed level of the output file. `<Value>` is a percentage between `0` and `1`, so 95% need to be given as `0.95`. No normalisation is done. The option only defines a level the volume may not exceed.
`/l=0.95`

3.5 The Section -azid()

Syntax
-azid([-g|--maximize] [-c] [-d] [-s] [-S] [-C] [-L] [-l] [-f] [-n] [-o] [--ch])

Azid decodes AC3 files and, if desired, performs a downmix from 6 channels. If BeSweet gets AC3 input but no -azid() section is defined, every option's default value will be used. For individual default values refer to the following pages. Default for the -d option is not fixed but depends on the number of channels expected by the destination format. For MP3 stereo downmix would always be used because MP3 does not support more than two channels. For Vorbis or AAC, which do support multichannel, -d's default is dependent on the options set in the respective encoder section.

3.5.1 Normalization and Dynamic Compression

-g {max|<level>}

Performs pre gain normalisation similar to OTA's -g and -norm. max is the same as --maximize further down. The file first gets search for highest possible gain value, which is applied afterwards (two-pass approach like OTA's -norm).

```
-g max
```

<Level> may be used in different ways. A value between 0% and 99% works similar to -g max, using not highest possible gain but given percentage of that value.

```
-g 97%
```

A negative number followed by db works similar to above percentages. Only now the applied value is highest possible gain minus given number of decibels.

```
-g -3.2db
```

A positive number followed by db raises the volume by a fixed number of decibels, that is not relative to highest possible gain.

```
-g 4db
```

Compared to OTA's -g Azid's -g is faster but a little less accurate. Furthermore, when using list files Azid normalises every source file individually, possibly leading to volume jumps. To avoid this you should always normalise with -ota() when using list files.

`--maximize`

Performs two-pass pre gain normalisation identical to `-g max`. The file first gets searched for highest possible gain value, which is applied afterwards.

`-c {none|light|normal|heavy|inverse}` Default: `none`

Compresses the source file's dynamic range, i. e. reducing the range between loudest and most silent sample. The function has five modes.

`none`	No dynamic compression.
`light`	Light compression. This mode's compression is 50 % (-6 dB) lighter than `normal`.
`normal`	Normal compression. Decoders in standalone player commonly use this mode.
`heavy`	Heavy compression. Useful for noisy environments or low quality hifi equipment.
`inverse`	Inversion of the light-mode, i. e. loud samples become even louder, silent samples become more silent.

3.5.2 Downmix Options

`-d <front>/<rear>` Default: see start of chapter

Defines for how many destination channels Azid should downmix. Important: This option's values ignore the LFE channel. For stereo downmixing the `-s` option is interesting, too. Values are given as `channels front/channels rear`. Valid are: `1/0`, `2/0`, `3/0`, `1/1`, `2/1`, `3/1`, `1/2`, `2/2` and `3/2`.

```
-d 2/0
```

`-s {stereo|dpl|surround|dplii|surround2}` Default: `dpl`

Controls the mode used for stereo downmixing. This option only has an effect, if `-d 2/0` is selected.

`stereo`	Normal stereo downmix.
`dpl`	Downmix compatible with Dolby Pro Logic.
`surround`	Same as `dpl`.
`dplii`	Downmix compatible with Dolby Pro Logic II.
`surround2`	Same as `dplii`.

`-S {<level>|BSI}`

Default: `BSI`

Specifies the level used for mixing the rear channels into left and right front channels. The value is given in decibels. `BSI` sets the level according the the values contained in the source AC3's BSI info field.

```
-S BSI
-S -1.5db
```

`-C {<level>|BSI}`

Default: `BSI`

Specifies the level used for mixing the center channel into left and right front channels. The value is given in decibels. `BSI` sets the level according the the values contained in the source AC3's BSI info field.

```
-C -2db
```

`-L <level>`

Default: `0db`

Specifies the level used for mixing the LFE channel (bass channel) into left and right front channels. The value is given in decibels.

```
-L -3db
```

`-l <level>`

Default: `0db`

Specifies the level used for mixing the source LFE channel into the destination LFE channel. The value is given in decibels.

```
-l -2.5db
```

`-f {0|1}`

Default: `0`

Filters the rear channels when downmixing to stereo. The filter is a 2nd order butterworth filter providing an increasing phase shift according to frequency. At 7 kHz it is 90 degrees and –3 dB. Major applications for this filter include ensuring a proper pro logic downmix and/or correcting phasing problems (washy sound) for weirdly mastered audio streams.

`-f 1` turns the filter on, `-f 0` or the switch missing turns it of.

-n {0|1} Default: 0

Activates “dialog normalisation reduction” (DNR). 5.1 AC3 files have a field in their BSI info stating how far below maximum level the dialog channel’s (front center) subjective loudness is. According to that value the -n switch changes the dialog level to –31 dB. Normalisation is is not affected by this because DNR is applied before any normalisation function.

DNR is only recommended for two scenarios. First: When you transcode several 5.1 AC3s with different dialog levels to one output file. Second: For equalising different dialog levels within a single source AC3 (possibly a TV capture with loud commercials). DNR has no effect when transcoding AC3 input without different dialog levels (should be the case for most DVD audio streams). Additionally AC3s without a center channel (e. g. 2/0) are not affected.

-n 1 turns the filter on, -n 0 or the switch missing turns it of.

3.5.3 Options for Controlling Individual Channels

These options are not needed for ordinary AC3 transcoding. -o becomes important, when changing channel order from input to output file. The --ch options should only be used by experienced users with a good reason for doing so.

-o <channel order>

Defines the output file’s channel order. This function is only needed when channel order for input and output file is different. Channels are given as abbreviations and separated by commas. Note that channel names refer to their position in the *input file*.

l	FL channel of the source (front left).
r	FR channel of the source (front right).
c	C channel of the source (front center).
lfe	LFE channel of the source.
sl	SL channel of the source (rear left).
sr	SR channel of the source (rear right).

Use this option, for example, to transcode 5.1 AC3 to 5.1 Vorbis. AC3 has a channel order of FL, FR, C, LFE, SL, SR while Vorbis uses FL, C, FR, SL, SR, LFE. For this example the -o option should look like this:

```
-o l,c,r,sl,sr,lfe
```

`--ch<number>={light|normal|heavy|inverse},<dB>`

Defines separate values for dynamic compression and gain for individual channels. `<Number>` stands for the channel's position (`0` to `5`) in the *output file*. So you must consider any changes done by `-o`. The first value (before the comma) sets dynamic compression mode. Same modes as for `-c` are used. The second value sets a fixed gain value in decibels for this individual channel. Write a `db` following the value.

Example: To used light compression and attenuate volume by 2.5 dB for channel number 4 (defaults to SL for AC3) use the option like this:

```
--ch4=light,-2.5db
```

`--ch<name>={light|normal|heavy|inverse},<dB>`

This option works similar to `--ch<number>`, but identifying channels by their position in the source. Abbreviations are the same as for `-o`.

Above example (SL channel with light compression and attenuation of 2.5 dB) now would look like this:

```
--chsl=light,-2.5db
```

4 Encoders

These sections must not be combined, meaning you can only use one of them in a single command line. If no encoder section is defined, you should use one of `-core()`'s destination format switches.

4.1 The Section `-ogg()`

Syntax
`-ogg( [-q|{-b [-m] [-M]}] [-6ch] )`

Encodes to Ogg Vorbis. Vorbis is primarily designed to use true VBR in constant quality mode. That is why you should always use the `-q` option to define output quality and bitrate.

The section may be defined empty as `-ogg()`. Then BeSweet will encode with an average bitrate of 160 kbit/s. (identical to `-ogg( -b 160 )`).

`-q <quality>`

Sets the quality level for encoding the file. Maximum allowed value is `1.000`. Minimum is `-0.200`.

By default Vorbis uses a scale from –2 to 10 that is different from BeSweet's default of –0.2 to 1. Converting is easily done by multiplying BeSweet's value by 10 or dividing the Vorbis value by 10. BeSweet's `0.250` corresponds to `2.50` for Vorbis and vice versa.

`-q 0.200`

For values below `0` BeSweet's screen output reads `Avarege Bitrate: 160` (including the typo), but actual transcoding is done with the correct value.

`-b <kbps>`

Sets nominal bitrate in kbit/s (ABR mode).

`-b 128`

-m <kbps>

Sets minimum allowed bitrate in kbit/s (also needs `-b`).
`-m 64`

-M <kbps>

Sets maximum allowed bitrate in kbit/s (also needs `-b`).
`-M 160`

-6ch

Creates a 6-channel Ogg Vorbis with channel order FL, C, FR, SL, SR, LFE. If the source's order is different, channels must be redirected accordingly.

This switch does not work with `-ssrc()`, `-boost()` und `-ota( -g )`.

4.2 The Section `-lame()`

LAME encodes MP3 files. Depending on the version of used *lame_enc.dll* different sets of parameters are recommended. Accordingly chapter 4.2.1 deals with LAME up to version 3.96 (usually not recommended any more) and chapter 4.2.2 covers LAME from version 3.97. Then chapter 4.2.3 discusses additional options only relevant under special circumstances.

The `-lame()` section may be defined empty as `-lame()`. Then BeSweet will encode to stereo with a constant bitrate of 128 kbit/s (identical to `-lame( --preset cbr 128 )`).

4.2.1 LAME up to Version 3.96

Syntax
`-lame( [--preset] [<more options>] )`

--preset {{[fast] medium|[fast] standard|[fast] extreme|insane}|cbr <kbps>|<kbps>}

Defines mode and quality for encoding the MP3. Constant quality is set by `cbr` followed by a value in kbit/s. Valid are `8`, `16`, `32`, `40`, `48`, `56`, `64`, `80`, `96`, `112`, `128`, `160`, `192`, `224`, `256` and `320`. Encoding to a constant bitrate of 192 kbit/s looks like this:

```
--preset cbr 192
```

Additionally the named preset `insane` uses a constant bitrate of 320 kbit/s. Writing `--preset insane` is the same as `--preset cbr 320`.

An average bitrate (ABR) is set simply by any number in kbit/s between 77 (for lame 3.90) or 72 (for lame 3.96) and 320. Encoding to an average bitrate of 135 kbit/s looks like this:

```
--preset 135
```

Variable bitrate is set by a named preset. From lowest to highest bitrate (= quality) they are called `medium`, `standard`, `extreme`. Every preset may be preceded by `fast` to use the new VBR mode, that is a little less accurate but a lot faster.

```
--preset standard
--preset fast medium
```

Instead of `--preset` you may also use `--alt-preset`.

```
--alt-preset standard
--alt-preset 150
```

4.2.2 LAME Version 3.97 and Newer

Syntax

```
-lame( -v --vbr-{old|new} -V [-b] [-B] [<more options>] )
-lame( {-b|--abr} [-b] [-B] [<more options>] )
```

BeSweet defaults to 128 kbit/s as minimal bitrate. Thus for low-quality VBR (from about `-V 5`) or low ABR bitrates (from about 140 kbit/s) lowering this value using the `-b` parameter is useful. Otherwise a final bitrate lower than 128 kbit/s is not possible.

-v

Activates VBR encoding. Additionally this parameters requires both `--vbr-old` or `--vbr-new` and `-V` in the same order as shown in the syntax above.

--vbr-{old|new}

Chooses the VBR mode, with `--vbr-old` being the older and slower standard VBR mode and `--vbr-new` being the faster new mode. Additionally this parameters requires both `-v` and `-V` in the same order as shown in the syntax above.

-V <quality>

Sets the quality level for VBR encoding. Possible values as integer numbers from `0` to `8`, where `0` stands for the highest, `8` for the lowest quality/bitrate.

Additionally this parameters requires both `-v` and `--vbr-old` or `--vbr-new` in the same order as shown in the syntax above.

```
-v --vbr-new -V 2
```

-b <kbps>

Uses CBR encoding with given bitrate in kbit/s. Valid are `8`, `16`, `32`, `40`, `48`, `56`, `64`, `80`, `96`, `112`, `128`, `160`, `192`, `224`, `256` and `320`.

```
-b 128
```

When used in ABR or VBR mode, this parameter defines the minimal possible bitrate for each audio frame. Valid are `32`, `40`, `48`, `56`, `64`, `80`, `96`, `112`, `128`, `160`, `192`, `224`, `256` and `320`. When using both `-b` and `-B`, minimum bitrate must always be lower than maximum bitrate.

```
-v --vbr-new -V 7 -b 32
--abr 128 -b 64
```

-B <kbps>

For ABR and VBR encoding sets maximum possible bitrate for each audio frame. Valid are `32`, `40`, `48`, `56`, `64`, `80`, `96`, `112`, `128`, `160`, `192`, `224`, `256` and `320`. When using both `-b` and `-B`, minimum bitrate must always be lower than maximum bitrate.

```
-v --vbr-old -V 3 -B 224
--abr 96 -b 32 -B 256
```

--abr <kbps>

Encodes with given average bitrate (ABR mode). Possible values for `<kbps>` range from `72` to `320`.

4.2.3 Special Options

Syntax
`-lame( <encoding mode> [--resample] [--lowpass] [-m] [-p] [-Y] )`

`--resample <Hz>`

Changes destination sampling frequency to the given value in Hertz. Real resampling should be done by `-shibatch()`. This option is there to prevent lame's automatic downsampling for low bitrates (below 88 kbit/s). Possible values for `<Hz>` are `8000`, `11025`, `12000`, `16000`, `22050`, `24000`, `32000`, `44100`, `48000`. To keep a source sampling rate of 44100 Hz for an encoding to below 88 kbit/s, the option would read:

```
--resample 441000
```

`--lowpass <kHz>`

Defines lowpass frequency in kilohertz. All frequencies above given value will be truncated. Maximum possible value is half the sampling frequency. When using a 48 kHz source that would be 24. Reasonable values are always lower because there cannot be frequencies above 24 kHz in an 48 kHz file.

```
--lowpass 18.5
```

Lame applies lowpass filters automatically, depending on chosen quality setting. Accordingly `--lowpass` should only be used under exceptional circumstances.

`-m {s|j|f|m}`

Default: `j`

This option is only necessary for downmixing to mono. Use it like this:

```
-m m
```

The other three modes set the method used for encoding stereo channels. The default `j` is fine for stereo encodings and should not be changed.

`s` normal stereo
`j` joint stereo
`f` forced joint stereo

`-p`

Stores a 16-bit CRC checksum for each audio frame (error protection). The checksum is part of the bitrate, i. e. less bits are available for the actual audio data.

-Y

Changes ATH mode to ignore noise in sfb21.

4.3 The Sections -mp2enc() and -toolame()

Syntax

```
-mp2enc( [-b|-v] [-m] [-e] )
-toolame( [-b|-v] [-m] [-e] )
```

By the name of this section you chose, which MP2 encoder BeSweet should use: MP2enc or 2Lame. Options are the same for both. The section may be defined empty as `-mp2enc()` or `-toolame()`. Then BeSweet will encode in dual channel mode using 192 kbit/s CBR.

-b <kbps> Default: 192

Uses CBR encoding with given bitrate in kbit/s.

```
-b 192
```

-v <quality>

Uses VBR encoding with given quality level. Possible are values between `0` and `9`. Warning: VBR mode is experimental and not guaranteed to work properly.

```
-v 5
```

-m {s|j|d|m} Default: d

Sets the stereo mode used. `s` for normal stereo, `j` for joint stereo, `d` for dual channel and `m` for mono.

```
-m s
```

-e

Adds a checksum to the stream as error protection.

4.4 The Section -ac3enc()

Syntax
```
-ac3enc( [-b] [-6ch] )
```

Encodes to AC3. Because of severe quality problems `-ac3enc()` should never be used. Aften is an alternative.

When using `-ac3enc()` normalisation must be done by Azid. OTA's normalisation settings will be ignored. The section may be defined empty as `-ac3enc()`. Then BeSweet will create a 6-channel AC3 with 384 kbit/s.

-b <kbps>

Default: 384

Sets bitrate in kbit/s. By default a video DVD uses 448 and 384 kbit/s for 6-channel AC3 and 192 kbit/s for stereo AC3.

```
-b 448
```

-6ch

Creates a 6-channel AC3. Without this switch set output channels will be the same as input channels.

4.5 The Section `-bsn()` for Aften AC3

Syntax
```
-bsn( -exe aften.exe {-b|-q} [-6chnew] )
```

Encodes to AC3 using the Aften encoder. The Option `-exe aften.exe` is required to activate Aften.

-b <kbps>

Encodes with the specified constant bitrate in CBR mode. Maximum bitrate is 640 kbit/s.

```
-b 384
```

-q <quality>

Encodes with the specified quality in VBR mode. Valid are integer numbers between `0` and `1023`.

```
-q 500
```

-6chnew

Tells Aften to create a 6-channel AC3. Without this option always a stereo file is produced. Aften does not respect the input's channel count.

4.6 The Section `-bsn()` for Flake FLAC

Syntax

```
-bsn( -exe flake.exe [-level] [-6chnew] )
```

Encodes to lossless FLAC using the Flake encoder. The option `-exe flake.exe` is required to activate Flake.

-level <level>

Specifies the compression level. Higher values produce smaller files but take more processing time. Valid levels are integer numbers between `0` and `12`.

```
-level 8
```

-6chnew

Tells Flake to create a 6-channel FLAC. Without this option always a stereo file is produced. Flake does not respect the input's channel count.

4.7 The Section `-bsn()` for Nero AAC

Syntax

```
-bsn( [-vbr|-abr|-cbr] [-aacprofile_{lc|he|hev2}] [-6chnew]
[-hintforstreaming] )
```

Encodes to AAC using the freely available Nero CLI encoder. The output is always muxed into an MP4 container.

AAC is primarily designed to use true VBR in constant quality mode. That is why you should always use the `-vbr` option to define output quality and bitrate.

`-vbr <quality>`

Encodes with the specified quality in VBR mode. Valid are numbers between `0.00` (low quality) and `1.00` (high quality). An exception is 6-channel HE-AAC with a maximum of `0.45`.

```
-vbr 0.25
```

`-abr <kbps>`

Encodes with the specified average bitrate in ABR mode. Valid are positive integer numbers.

```
-abr 80
```

`-cbr <kbps>`

Encodes with the specified constant bitrate in CBR mode. Valid are positive integer numbers.

```
-cbr 100
```

`-aacprofile_{lc|he|hev2}` Default: Automatic

Forces encoding with the specified AAC profile. `-aacprofile_hev2` is only possible for stereo output and designed for extremely low bitrates. LC and HE work with both stereo and 6-channel data.

`lc`	LC (Low Complexity).
`he`	HE (High Efficiency) = LC + Spectral Band Replication (SBR)
`hev2`	HE v2 = LC + SBR + Parametric Stereo (PS)

If this option is absent Nero chooses the AAC profile automatically depending on quality/bitrate and channel count. The following table shows the values used by Nero 1.1.34.2.

	`-vbr`	`-abr` and `-cbr`
stereo output	Till 0.15 HE v2, till 0.30 HE, above LC.	Till 39 kbit/s HE v2, till 84 kbit/s HE, above LC.
6-channel output	Till 0.30 HE, above LC.	Till 254 kbit/s HE, above LC.

`-6chnew`

Tells Nero to create a 6-channel AAC. Without this option always a stereo file is produced. Nero does not respect the input's channel count.

`-hintforstreaming`

Adds hinting used on streaming servers.

4.8 The Section `-dimzon()` for Winamp AAC

Syntax

```
-dimzon( -dllname WA_aac.dll -cbr [-lc] [-high] [-mode]
[-mpeg2] [-pns] [-6chnew] )
```

Encodes to AAC using the Winamp encoder. The output file is a raw AAC stream, not muxed into MP4. Only CBR encoding is supported. The option `-dllname WA_aac.dll` is always required to activate Winamp.

`-cbr <kbps>`

Defines the output file's bitrate. Valid are integer numbers. Minimum and maximum depend on encoding mode and channel count according to the following table.

mode	stereo	6-channel
LC	96–256 kbit/s	160–320 kbit/s
HE	16–256 kbit/s	96–213 kbit/s
HE High	96–256 kbit/s	stereo only
HE + PS	16–48 kbit/s	stereo only

`-lc`

Activates the AAC profile *Low Complexity* (LC). Without this switch Winamp encodes in HE (High Efficiency) mode.

`-high`

Enables the HE-AAC encoder tuned for high bitrates. Works with stereo only.

`-mode {1|2|3|4|5}`

Defines the encoding mode for mono and stereo output.

1 Mono.
2 Joint stereo.
3 Normal stereo.
4 Parametric stereo (PS). Only together with HE-AAC.
5 Dual channel.

`-mpeg2`

Creates an MPEG-2 AAC bitstream. Without this switch MPEG-4 AAC is used.

`-pns`

Activates Perceptual Noise Substitution (PNS).

`-6chnew`

Tells Winamp to create a 6-channel AAC. Without this option always a stereo file is produced. Winamp does not respect the input's channel count.