Cheatography

CamillaDSP Cheat Sheet

by eweitzman via cheatography.com/214563/cs/46734/

top level

title:	Configuration name.
description:	Configuration description.
devices:	Defines devices for "capture" and "playback" (input and output devices).
mixers:	Mixers define channels, route audio, and change the number of channels in the pipeline.
filters:	Define and configure filters used in the pipeline.
proces-sors:	Special "filters" that work on several channels at once..
pipeline:	The processing steps to be followed.

devices:

sample-rate:	The samplerate setting decides the sample rate that everything will run at. This rate must be supported by both the capture and playback device.
chunksize:	All processing is done in chunks of data. The chunksize is the number of samples each chunk will have per channel. Suggested starting points for different sample rates: 44.1 or 48 kHz: 1024 88.2 or 96 kHz: 2048 176.4 or 192 kHz: 4096 The duration in seconds of a chunk is chunksize/sa-mplerate, so the suggested values corresponds to about 22 ms per chunk. This is a reasonable value.

devices: (cont)

queuelimit:*	[defaults to 4] The field queuelimit should normally be left out to use the default of 4. It sets the limit for the length of the queues between the capture device and the processing thread, as well as between the processing thread and the playback device. The total queue size limit will be 2 * *chunksize* * *queuelimit* samples per channel.
enable_rate_adj-ust:*	[default *false*] This enables the playback device to control the rate of the capture device, in order to avoid buffer underruns or a slowly increasing latency. This is currently supported when using an Alsa, Wasapi or CoreAudio playback device (and any capture device).
target_level:*	[default *chunksize*] The value is the number of samples that should be left in the buffer of the playback device when the next chunk arrives. Only applies when *enable_rate_adjust* is set to true.
adjust_period:*	[defaults 10] Set the interval between correc-tions, in seconds. Only applies when *enable-_rate_adjust* is set to true
silence_threshold:* silence_timeout:*	Pause processing if the input is silent. The threshold is the threshold level in dB.
capture_sampler-ate:*	The capture samplerate. Setting it to null sets the capture samplerate to the same value as samplerate.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

devices: (cont)

stop_on_rate_change:* rate_measure_interval:*	Setting *stop_on_rate_change* to true makes CamillaDSP stop the processing if the measured capture sample rate changes.
volume_ramp_time:*	[default 400 ms] Set the duration of the ramp when changing volume of the default volume control.
multithreaded:* worker_threads:*	[defaults to false and automatic)] Setting true enables multithreaded processing. When this is enabled, CamillaDSP creates several filtering tasks by grouping the filters for each channel.
capture: playback:	Defines input and output devices respectively. See the dedicated section.
resampler:*	Defines the resampler. Setting it to null or leaving it out disables resampling. See the dedicated section below.

(devices:) capture: and playback:

type:	Type type of device. See *Note1* below.
channels:	Number of channels.
device:	Device name (for Alsa, Pulse, Wasapi, CoreAudio). For CoreAudio and Wasapi, null will give the default device
filename:	Path to the file (for File, RawFile and WavFile).
format:	Sample format (for all except Jack).

Note1 - List of device types for the *type:* node.
All devices: Jack, Wasapi, CoreAudio, Alsa, Pulse
Capture devices only: Bluez, RawFile, WavFile, SignalGenerator, and Stdin
Playback devices only: File and Stdout.
Note2 - *capture:* device types have an optional *labels:* property. This accepts a list of strings, and is meant to be used by a GUI to display meaningful channel names. CamillaDSP itself does not use these labels.

(devices:) resampler:

type:	One of AsyncSinc, AsyncPoly, or Synchronous.
AsyncSinc resampling	Asynchronous resampling with anti-aliasing.
profile: (AsyncSinc)	One of Balanced, Fast, VeryFast, or Accurate
more AsyncSinc parameters	See the CamillaDSP resampling documentation or the Rubato documentation.
AsyncPoly resampling	Asynchronous resampling without anti-aliasing.
interpolation: (AsyncPoly)	One of Linear, Cubic, Quintic, or Septic.
Synchronous resampling	Synchronous resampling with anti-aliasing.

The Synchronous resampler has no additional nodes.

mixer:

Each child node of the *mixer:* node is a user-defined mixer name that will be referenced in the ____ node(s). Each mixer has the following nodes. See the CamillaDSP documentation for more info.

description:*			Mixer description.
labels:*			Labels for GUI display..
channels:			
	in:		Number of input channels.
	out:		Number of output channels.
mapping:			A list containing one *dest* entry for each output channel. Channel numbers are 0-based.
	dest:		An output channel.
	mute:*		Should the output channel be muted? Boolean.
	sources:		A list of sources contributing to the output channel.
		channel:	Input source channel number. 0-based.
		gain:*	Gain to apply to the input channel. Units are in *scale*.
		scale:*	Units for *gain*. Must be one of *dB* or *linear*.
		inverted:*	Should the channel be inverted? Boolean.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Cheatography

CamillaDSP Cheat Sheet

by eweitzman via cheatography.com/214563/cs/46734/

processors:

Each node under the *processors:* node is a user-defined name for a configured processor.

type:	A processor type, one of Compressor or NoiseGate.
parameters:	See Compressor parameters and NoiseGate parameters
Compressor	A standard dynamic range compressor configured with the most common parameters.
NoiseGate	Simple noise gate that estimates current loudness using the compressor's algorithm.

pipeline:

The pipeline consists of a list of processing steps between input and output. Each step has a type , affected channels, and the name(s) of each filter, mixer, or processor.

type:	One of Mixer, Filter, or Processor.
description*:	[blank] The pipeline description.
name:	For Mixer or Processor steps only. User-defined name for a mixer or processor to use..
names:	For Filter steps only. List of user-defined names of filters to use.
channels:	For Filter steps only. List of channels to filter.
bypassed*:	[false] Bypasses the processing step if *true*.

If the name of a mixer, processor or filter includes the tokens $samplerate$ or $channels$, these will be replaced by the corresponding values from the config. For example, if the samplerate is 44100, the filter name *fir_$samplerate$* will be updated to *fir_44100*.

filters: Utility filters

Gain		Gain to apply.
	gain:	Range is +/-150 dB or +/- 10 (linear scale).
	scale*:	[dB] *dB* or *linear*.
	mute*:	[false] Mute the signal.
	inverted*:	[false] Invert the signal.

filters: Utility filters (cont)

Volume		Volume control that reacts to a websocket request to change the volume of a fader. All Volume filters responding to the fader in the request will change their volume.
	fader:	Which fader (`Aux1` to `Aux4`) that this filter responds to.
	limit*:	[+50 dB] Maximum value allowed for the filter.
	ramp_time*:	[400 ms] Time interval (in milliseconds) over which the volume changes to a new value. Must be positive.
Loudness		Apply loudness compensation using the RME ADI-2 DAC FS method. Reacts to changes in volume for any of the five faders. Can be used with an external volume control if changes to that control are also sent to CamillaDSP's websocket.
	reference_level:	If the fader is above this level, no compensation is applied. Below 20 dB below the *reference_level*, the full amount of each boost is applied. Otherwise, the boosts are scaled linearly. Range is +20/-100.
	fader*:	[Main] The fader whose volume is monitored to determine the amount of compensation to apply.
	high_boost*:	[10 dB] Maximum HF boost in dB. Range is 0-20.
	low_boost*:	[10 dB] Maximum LF boost in dB. Range is 0-20.
	attenuate_mid*:	[false] Attenuates mids instead of boosting HF and LF to avoid clipping when used with external volume controls and web API.
Delay		Delays the signal. If *unit:* is *mm*, the speed of sound of 343 m/sec is used to calculate the delay.

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

filters: Utility filters (cont)

	delay:	Amount of delay. Must be zero or positive.
	unit*:	[ms] *delay:* units, one of *ms*, *mm*, or *samples*.
	subsample:	When *true*, an IIR filter is used to achieve subsample delay precision. When *false*, will round to the nearest number of full samples. This is faster.
Dither		Intended to add shaped noise to 16-bit output. Several types are supported
	type:	Type of dither. See **documentation**.
	bits:	Target bit depth. Should match bitdepth of oversampling delta-sigma DACs. For true NOS DACs, should match the number of bits where the DAC is linear.
Limiter		Limits output signal to a given level.
	clip_limit:	Clipping level in dB.
	soft_clip*:	[false] Enable soft clipping, introducing some harmonic distortion..
DiffEq		Implements a generic difference equation. See documentation.
	a:	[1.0] Coefficients a_0, a_1, ..., a_n.
	b:	[1.0] Coefficients b_0, b_1, ..., b_n.

filters: Biquad filters (IIR)

Many standard filters are implemented as Biquad IIR filters. These are specified using another *type:* node beneath the biquad's *parameters:* node. Values for this second *type:* node follow. Type-specific nodes define each parameter for that type and are shown below.
Note: *bandwidth* is given in octaves.

Free	a1: a2: b0: b1: b2:	Normalized coefficients.
Highpass Lowpass	freq: q:	Second order high/lowpass filters (12dB/oct).

filters: Biquad filters (IIR) (cont)

HighpassFO LowpassFO	freq:	First order high/lowpass filters (6dB/oct)
Highshelf Lowshelf	freq: gain: q: or slope:	*freq:* is the center frequency, where the gain is half of *grain:*. *slope:* is in dB/octave.
HighshelfFO LowshelfFO	freq: gain:	*freq:* is the center frequency, where the gain is half of *grain:*.
Peaking	freq: gain: q: or bandwidth:	
Notch	freq: q: or bandwidth:	Has a large negative gain.
Genera-lNotch	freq_z: freq_p: q_p: normalize_at_dc:*	A notch filter where the pole and zero can be placed at different frequencies. *q_p* is the pole Q.
Bandpass	freq: q: or bandwidth:	Second order bandpass filter.
Allpass	freq: q: or bandwidth:	Second order allpass filter.
AllpassFO	freq:	First order allpass filter.
LinkwitzT-ransform	freq_act: q_act: freq_target: q_target:	Modifies a given actual speaker HP response (frequency and q) with a new target.

filters: BiquadCombo filters (IIR)

These filters are composed of several Biquad IIR filters. These are specified using another *type:* node beneath the biquad's *parameters:* node. Values for this second *type:* node follow. Type-specific nodes define each parameter for that type and are shown below.
Note1: *bandwidth* is given in octaves.
Note2: See also Higher order biquads

ButterworthHighpass ButterworthLowpass	freq: order:

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com

Cheatography

CamillaDSP Cheat Sheet

by eweitzman via cheatography.com/214563/cs/46734/

filters: BiquadCombo filters (IIR) (cont)

LinkwitzRileyHighpass LinkwitzRileyLowpass	freq: order:	*order:* must be even.
Tilt	gain:	Tilts across entire spectrum. Positive values increase the high frequencies. Half the gain is applied to the highs and negative half is applied to the lows. Gain limited to +/-100dB
FivePointPeq		Five band PEQ with a low shelf, three peaking filters, and a high shelf. All 15 values are required. fls: gls: qls: fp1: gp1: qp1: fp2: gp2: qp2: fp3: gp3: qp3: fhs: ghs: qhs:
GraphicEqualizer	freq_min: freq_max: gains:	Min/max frequencies default to 20Hz and 20kHz. *gains:* is an array of gains for each band. The number of bands is set by the length of the array.

filters: Convolution filters (FIR)

Conv (FIR)

Volume control faders

There are five volume control "channels" called "faders", named "-Main" and "Aux1" to "Aux4". See volume control.

Supported sample formats

S16LE	Signed 16-bit int, stored as two bytes
S24LE	Signed 24-bit int, stored as four bytes (three bytes of data, one padding byte)
S24LE3	Signed 24-bit int, stored as three bytes (with no padding)
S32LE	Signed 32-bit int, stored as four bytes
FLOAT32LE	32-bit float, stored as four bytes
FLOAT64LE	64-bit float, stored as eight bytes

Supported sample formats by device type

	Alsa	Pulse	Wasapi	CoreAudio	Jack	Other
S16LE	Yes	Yes	Yes	Yes	No	Yes
S24LE	Yes	Yes	Yes	Yes	No	Yes
S24LE3	Yes	Yes	Yes	Yes	No	Yes
S32LE	Yes	Yes	Yes	Yes	No	Yes
FLOAT32LE	Yes	Yes	Yes	Yes	Yes	Yes
FLOAT64LE	Yes	No	No	No	No	Yes

Other = File/Stdin/Stdout

Equivalent sample formats by API

CamillaDSP	Alsa	Pulse
S16LE	S16_LE	S16LE
S24LE	S24_LE	S24_32LE
S24LE3	S24_3LE	S24LE
S32LE	S32_LE	S32LE
FLOAT32LE	FLOAT_LE	FLOAT32LE
FLOAT64LE	FLOAT64_LE	-

Sponsored by **Readable.com**
Measure your website readability!
https://readable.com