

# SAC Effects

v1.0

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Hierarchical Index</b>	<b>3</b>
2.1	Class Hierarchy . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class List . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Namespace Documentation</b>	<b>9</b>
5.1	Ui Namespace Reference . . . . .	9
<b>6</b>	<b>Class Documentation</b>	<b>11</b>
6.1	AudioInfo Class Reference . . . . .	11
6.1.1	Detailed Description . . . . .	12
6.1.2	Constructor & Destructor Documentation . . . . .	12
6.1.2.1	AudioInfo() . . . . .	12
6.1.3	Member Function Documentation . . . . .	12
6.1.3.1	setFile() . . . . .	13
6.2	AudioOutput Class Reference . . . . .	13
6.2.1	Detailed Description . . . . .	15
6.2.2	Constructor & Destructor Documentation . . . . .	15
6.2.2.1	AudioOutput() . . . . .	15

6.2.3	Member Function Documentation	15
6.2.3.1	setDevice	16
6.2.3.2	setDevices()	16
6.2.3.3	setFormat()	17
6.2.3.4	setVolume()	17
6.2.4	Member Data Documentation	18
6.2.4.1	fs	18
6.2.4.2	outputdevice	18
6.2.4.3	samplesize	18
6.2.4.4	volume	18
6.3	AudioSignal Class Reference	19
6.3.1	Detailed Description	20
6.3.2	Constructor & Destructor Documentation	20
6.3.2.1	AudioSignal() [1/2]	20
6.3.2.2	AudioSignal() [2/2]	20
6.3.3	Member Function Documentation	21
6.3.3.1	addSample()	21
6.3.3.2	deleteSample() [1/2]	21
6.3.3.3	deleteSample() [2/2]	22
6.3.3.4	getFrequencies() [1/2]	22
6.3.3.5	getFrequencies() [2/2]	22
6.3.3.6	getSample()	23
6.3.3.7	getSignal()	24
6.3.3.8	getSpectrum() [1/2]	24
6.3.3.9	getSpectrum() [2/2]	24
6.3.3.10	getTimes() [1/2]	25
6.3.3.11	getTimes() [2/2]	25
6.3.3.12	operator[]()	25
6.3.3.13	setSample()	26
6.3.3.14	setSignal()	27

6.3.4	Member Data Documentation . . . . .	27
6.3.4.1	fs . . . . .	27
6.3.4.2	maxsamples . . . . .	27
6.3.4.3	size . . . . .	27
6.4	AudioTest Class Reference . . . . .	28
6.4.1	Detailed Description . . . . .	28
6.4.2	Constructor & Destructor Documentation . . . . .	29
6.4.2.1	AudioTest() . . . . .	29
6.5	BinauralQuality Struct Reference . . . . .	29
6.5.1	Detailed Description . . . . .	29
6.5.2	Member Enumeration Documentation . . . . .	30
6.5.2.1	binauralquality . . . . .	30
6.6	Channel Class Reference . . . . .	31
6.6.1	Detailed Description . . . . .	32
6.6.2	Constructor & Destructor Documentation . . . . .	32
6.6.2.1	Channel() . . . . .	32
6.6.3	Member Function Documentation . . . . .	33
6.6.3.1	bypass() . . . . .	33
6.6.3.2	getIndex() . . . . .	33
6.6.3.3	mute() . . . . .	34
6.6.3.4	setLabel() . . . . .	34
6.6.3.5	setVolume() . . . . .	34
6.6.4	Member Data Documentation . . . . .	34
6.6.4.1	audiooutput . . . . .	34
6.6.4.2	bypasscheckbox . . . . .	35
6.6.4.3	bypassed . . . . .	35
6.6.4.4	deviceselector . . . . .	35
6.6.4.5	groupbox . . . . .	35
6.6.4.6	index . . . . .	35
6.6.4.7	label . . . . .	36

6.6.4.8	<a href="#">mutecheckbox</a>	36
6.6.4.9	<a href="#">muted</a>	36
6.6.4.10	<a href="#">name</a>	36
6.6.4.11	<a href="#">volume</a>	36
6.6.4.12	<a href="#">volumeslider</a>	37
6.7	<a href="#">ChannelsCharts Class Reference</a>	37
6.7.1	<a href="#">Detailed Description</a>	38
6.7.2	<a href="#">Constructor &amp; Destructor Documentation</a>	38
6.7.2.1	<a href="#">ChannelsCharts()</a>	38
6.8	<a href="#">ChannelsList Class Reference</a>	38
6.8.1	<a href="#">Detailed Description</a>	40
6.8.2	<a href="#">Constructor &amp; Destructor Documentation</a>	40
6.8.2.1	<a href="#">ChannelsList()</a>	40
6.8.3	<a href="#">Member Function Documentation</a>	40
6.8.3.1	<a href="#">deleteChannel()</a>	41
6.8.3.2	<a href="#">getChannel()</a>	41
6.8.3.3	<a href="#">getNames()</a>	42
6.8.3.4	<a href="#">getSize()</a>	43
6.8.3.5	<a href="#">setSize()</a>	43
6.8.4	<a href="#">Member Data Documentation</a>	44
6.8.4.1	<a href="#">fs</a>	44
6.8.4.2	<a href="#">samplesize</a>	44
6.9	<a href="#">SACBitstream::ChannelType Struct Reference</a>	44
6.9.1	<a href="#">Detailed Description</a>	45
6.9.2	<a href="#">Member Enumeration Documentation</a>	45
6.9.2.1	<a href="#">channeltype</a>	45
6.10	<a href="#">Chart2D Class Reference</a>	45
6.10.1	<a href="#">Detailed Description</a>	47
6.10.2	<a href="#">Constructor &amp; Destructor Documentation</a>	47
6.10.2.1	<a href="#">Chart2D()</a> [1/2]	47

6.10.2.2	<a href="#">Chart2D()</a> [2/2]	47
6.10.3	<a href="#">Member Function Documentation</a>	48
6.10.3.1	<a href="#">getPoints()</a>	48
6.10.3.2	<a href="#">setOptions()</a>	48
6.10.3.3	<a href="#">setPoints()</a>	49
6.10.3.4	<a href="#">setRange()</a>	50
6.10.3.5	<a href="#">setTitle()</a>	50
6.10.4	<a href="#">Member Data Documentation</a>	51
6.10.4.1	<a href="#">xlabel</a>	51
6.10.4.2	<a href="#">ylabel</a>	51
6.11	<a href="#">Chart2D::ChartOptions Struct Reference</a>	51
6.11.1	<a href="#">Detailed Description</a>	51
6.11.2	<a href="#">Member Enumeration Documentation</a>	51
6.11.2.1	<a href="#">Options</a>	51
6.12	<a href="#">Compressor Class Reference</a>	52
6.12.1	<a href="#">Detailed Description</a>	52
6.12.2	<a href="#">Member Function Documentation</a>	52
6.12.2.1	<a href="#">apply()</a>	52
6.12.2.2	<a href="#">plot()</a>	53
6.13	<a href="#">DecodingType Struct Reference</a>	54
6.13.1	<a href="#">Detailed Description</a>	54
6.13.2	<a href="#">Member Enumeration Documentation</a>	54
6.13.2.1	<a href="#">decodingtype</a>	54
6.14	<a href="#">Effect Class Reference</a>	55
6.14.1	<a href="#">Detailed Description</a>	56
6.14.2	<a href="#">Member Enumeration Documentation</a>	56
6.14.2.1	<a href="#">effectID</a>	56
6.14.3	<a href="#">Constructor &amp; Destructor Documentation</a>	56
6.14.3.1	<a href="#">Effect()</a> [1/2]	56
6.14.3.2	<a href="#">Effect()</a> [2/2]	57

6.14.4	Member Function Documentation	57
6.14.4.1	apply()	57
6.14.4.2	getChannels()	58
6.14.4.3	getEffect()	59
6.14.4.4	getEffects()	60
6.14.4.5	getLevels()	61
6.14.4.6	getParams()	62
6.14.4.7	getTag()	63
6.14.4.8	getTagMap()	63
6.14.4.9	plot()	64
6.14.4.10	setParams()	65
6.14.5	Member Data Documentation	66
6.14.5.1	effect	66
6.15	EffectBase Class Reference	66
6.15.1	Detailed Description	67
6.15.2	Member Function Documentation	67
6.15.2.1	getBool()	67
6.15.2.2	getDouble()	67
6.15.2.3	getInt()	68
6.15.3	Member Data Documentation	68
6.15.3.1	fs	68
6.15.3.2	params	68
6.16	EffectsMonitor Class Reference	69
6.16.1	Detailed Description	70
6.16.2	Constructor & Destructor Documentation	70
6.16.2.1	EffectsMonitor() [1/2]	70
6.16.2.2	EffectsMonitor() [2/2]	71
6.16.3	Member Function Documentation	71
6.16.3.1	setEffect()	71
6.16.3.2	setParameter()	72



6.16.3.3	<a href="#">updateParameter [1/4]</a>	73
6.16.3.4	<a href="#">updateParameter [2/4]</a>	73
6.16.3.5	<a href="#">updateParameter [3/4]</a>	74
6.16.3.6	<a href="#">updateParameter [4/4]</a>	74
6.16.4	<a href="#">Member Data Documentation</a>	75
6.16.4.1	<a href="#">charts</a>	75
6.16.4.2	<a href="#">effect</a>	75
6.16.4.3	<a href="#">effects</a>	75
6.16.4.4	<a href="#">files</a>	76
6.16.4.5	<a href="#">parameters</a>	76
6.17	<a href="#">Encoder Class Reference</a>	76
6.17.1	<a href="#">Detailed Description</a>	77
6.17.2	<a href="#">Member Function Documentation</a>	78
6.17.2.1	<a href="#">setInput()</a>	78
6.17.2.2	<a href="#">setOutput()</a>	78
6.17.2.3	<a href="#">setTree()</a>	79
6.17.3	<a href="#">Member Data Documentation</a>	80
6.17.3.1	<a href="#">bitstream</a>	80
6.17.3.2	<a href="#">fs</a>	80
6.17.3.3	<a href="#">input</a>	80
6.17.3.4	<a href="#">output</a>	81
6.18	<a href="#">File::Endianness Struct Reference</a>	81
6.18.1	<a href="#">Detailed Description</a>	81
6.18.2	<a href="#">Member Enumeration Documentation</a>	81
6.18.2.1	<a href="#">endianness</a>	81
6.19	<a href="#">Equalizer Class Reference</a>	81
6.19.1	<a href="#">Detailed Description</a>	82
6.19.2	<a href="#">Member Function Documentation</a>	82
6.19.2.1	<a href="#">apply()</a>	82
6.19.2.2	<a href="#">filter()</a>	83

6.19.2.3	highShelfFilter()	84
6.19.2.4	lowShelfFilter()	84
6.19.2.5	peakingFilter()	85
6.20	File Class Reference	86
6.20.1	Detailed Description	87
6.20.2	Constructor & Destructor Documentation	87
6.20.2.1	File()	87
6.20.3	Member Function Documentation	88
6.20.3.1	exists()	88
6.20.3.2	getCursor()	89
6.20.3.3	getFilename()	89
6.20.3.4	read()	89
6.20.3.5	readNumber()	90
6.20.3.6	readText()	91
6.20.3.7	setCursor()	92
6.20.3.8	setFilename()	93
6.20.3.9	size()	93
6.20.3.10	write()	94
6.20.3.11	writeNumber()	95
6.20.3.12	writeText()	95
6.21	WAVFile::Header Struct Reference	96
6.21.1	Detailed Description	97
6.21.2	Member Data Documentation	97
6.21.2.1	audioformat	97
6.21.2.2	bitspersample	97
6.21.2.3	blockalign	98
6.21.2.4	byterate	98
6.21.2.5	chunkID	98
6.21.2.6	chunksize	98
6.21.2.7	format	98

6.21.2.8	numchannels	99
6.21.2.9	samplerate	99
6.21.2.10	subchunk1ID	99
6.21.2.11	subchunk1size	99
6.21.2.12	subchunk2ID	99
6.21.2.13	subchunk2size	100
6.22	HRTFModel Struct Reference	100
6.22.1	Detailed Description	100
6.22.2	Member Enumeration Documentation	100
6.22.2.1	hrtfmodel	100
6.23	LogType Struct Reference	101
6.23.1	Detailed Description	101
6.23.2	Member Enumeration Documentation	101
6.23.2.1	logtype	101
6.24	OutputDevice Class Reference	101
6.24.1	Detailed Description	103
6.24.2	Constructor & Destructor Documentation	103
6.24.2.1	OutputDevice()	103
6.24.3	Member Function Documentation	103
6.24.3.1	bytesAvailable()	104
6.24.3.2	readData()	104
6.24.3.3	send()	104
6.24.3.4	test()	105
6.24.3.5	writeData()	106
6.24.4	Member Data Documentation	106
6.24.4.1	buffer	106
6.24.4.2	buffersize	106
6.24.4.3	cursor_read	106
6.24.4.4	cursor_write	107
6.25	Panning Class Reference	107

6.25.1 Detailed Description . . . . .	107
6.25.2 Member Function Documentation . . . . .	107
6.25.2.1 apply() . . . . .	107
6.26 ProcessManager Class Reference . . . . .	108
6.26.1 Detailed Description . . . . .	109
6.26.2 Constructor & Destructor Documentation . . . . .	109
6.26.2.1 ProcessManager() . . . . .	109
6.26.3 Member Function Documentation . . . . .	110
6.26.3.1 applyEffect() . . . . .	110
6.26.3.2 decode() . . . . .	111
6.26.3.3 setInput() . . . . .	112
6.26.3.4 setOutput() . . . . .	112
6.26.4 Member Data Documentation . . . . .	113
6.26.4.1 channels . . . . .	113
6.26.4.2 cursor . . . . .	113
6.26.4.3 fs . . . . .	114
6.26.4.4 input . . . . .	114
6.26.4.5 NRG . . . . .	114
6.26.4.6 OLD . . . . .	114
6.26.4.7 output . . . . .	114
6.26.4.8 samples . . . . .	115
6.26.4.9 total . . . . .	115
6.27 Reverb Class Reference . . . . .	115
6.27.1 Detailed Description . . . . .	116
6.27.2 Member Function Documentation . . . . .	116
6.27.2.1 apply() . . . . .	116
6.27.2.2 combfilter() . . . . .	117
6.27.2.3 feedforwardfilter() . . . . .	118
6.27.2.4 lowpassfeedbackfilter() . . . . .	118
6.27.2.5 schroederdiffusionfilter() . . . . .	119

6.27.2.6	<a href="#">schroederfilter()</a>	120
6.28	<a href="#">SACBitstream Class Reference</a>	121
6.28.1	<a href="#">Detailed Description</a>	122
6.28.2	<a href="#">Member Function Documentation</a>	122
6.28.2.1	<a href="#">getVariable()</a>	122
6.28.3	<a href="#">Member Data Documentation</a>	123
6.28.3.1	<a href="#">channels</a>	123
6.28.3.2	<a href="#">fs</a>	123
6.28.3.3	<a href="#">gain_downmix</a>	123
6.28.3.4	<a href="#">gain_LFE</a>	123
6.28.3.5	<a href="#">gain_surround</a>	124
6.29	<a href="#">SACEffects Class Reference</a>	124
6.29.1	<a href="#">Detailed Description</a>	125
6.29.2	<a href="#">Constructor &amp; Destructor Documentation</a>	126
6.29.2.1	<a href="#">SACEffects()</a>	126
6.29.3	<a href="#">Member Function Documentation</a>	126
6.29.3.1	<a href="#">setBinauralQuality()</a>	126
6.29.3.2	<a href="#">setBitstream()</a>	127
6.29.3.3	<a href="#">setDecodingType()</a>	128
6.29.3.4	<a href="#">setDuration()</a>	128
6.29.3.5	<a href="#">setEffect()</a>	129
6.29.3.6	<a href="#">setFormat()</a>	129
6.29.3.7	<a href="#">setHRTFModel()</a>	130
6.29.3.8	<a href="#">setInput()</a>	131
6.29.3.9	<a href="#">setSource()</a>	132
6.29.3.10	<a href="#">setUpmixType()</a>	133
6.29.4	<a href="#">Member Data Documentation</a>	134
6.29.4.1	<a href="#">fs</a>	134
6.30	<a href="#">UpmixType Struct Reference</a>	134
6.30.1	<a href="#">Detailed Description</a>	134

6.30.2	Member Enumeration Documentation	134
6.30.2.1	upmixtype	134
6.31	WAVFile Class Reference	135
6.31.1	Detailed Description	136
6.31.2	Constructor & Destructor Documentation	136
6.31.2.1	WAVFile() [1/3]	136
6.31.2.2	WAVFile() [2/3]	137
6.31.2.3	WAVFile() [3/3]	137
6.31.3	Member Function Documentation	138
6.31.3.1	getCursor()	138
6.31.3.2	readSamples()	138
6.31.3.3	samples()	139
6.31.3.4	setCursor()	140
6.31.3.5	writeSamples()	140
6.31.4	Member Data Documentation	141
6.31.4.1	duration	141
6.31.4.2	header	141
<b>7</b>	<b>File Documentation</b>	<b>143</b>
7.1	src/tools/Logger.cpp File Reference	143
7.1.1	Detailed Description	144
7.1.2	Function Documentation	144
7.1.2.1	consolelog()	144
7.1.2.2	tab()	145
7.1.3	Variable Documentation	146
7.1.3.1	black	146
7.1.3.2	blue	147
7.1.3.3	bold	147
7.1.3.4	cyan	147
7.1.3.5	green	147
7.1.3.6	grey	148
7.1.3.7	italic	148
7.1.3.8	magenta	148
7.1.3.9	red	148
7.1.3.10	reset	148
7.1.3.11	yellow	148

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Ui

Qt user interfaces namespace (see <http://doc.qt.io/qt-5/topics-ui.html>) . . . 9





## Chapter 2

# Hierarchical Index

### 2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

AudioSignal . . . . .	19
BinauralQuality . . . . .	29
Channel . . . . .	31
SACBitstream::ChannelType . . . . .	44
Chart2D::ChartOptions . . . . .	51
Compressor . . . . .	52
DecodingType . . . . .	54
Effect . . . . .	55
EffectBase . . . . .	66
File::Endianness . . . . .	81
Equalizer . . . . .	81
File . . . . .	86
WAVFile . . . . .	135
WAVFile::Header . . . . .	96
HRTFModel . . . . .	100
LogType . . . . .	101
Panning . . . . .	107
ProcessManager . . . . .	108
QDialog	
AudioInfo . . . . .	11
AudioTest . . . . .	28
ChannelsCharts . . . . .	37
Encoder . . . . .	76
QIODevice	
OutputDevice . . . . .	101
QMainWindow	
SACEffects . . . . .	124
QObject	
AudioOutput . . . . .	13
ChannelsList . . . . .	38
EffectsMonitor . . . . .	69
QWidget	
Chart2D . . . . .	45
Reverb . . . . .	115
SACBitstream . . . . .	121
UpmixType . . . . .	134



## Chapter 3

# Class Index

### 3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

<a href="#">AudioInfo</a>	Audio object info dialog class . . . . .	11
<a href="#">AudioOutput</a>	Audio output class . . . . .	13
<a href="#">AudioSignal</a>	Audio signal class . . . . .	19
<a href="#">AudioTest</a>	Audio output test class . . . . .	28
<a href="#">BinauralQuality</a>	SAC decoder parameter binaural quality . . . . .	29
<a href="#">Channel</a>	Single-object class from channels list . . . . .	31
<a href="#">ChannelsCharts</a>	. . . . .	37
<a href="#">ChannelsList</a>	Channels list class. It shows information about channels signals . . . . .	38
<a href="#">SACBitstream::ChannelType</a>	It specifies the channel type . . . . .	44
<a href="#">Chart2D</a>	Class for plotting two-dimensional charts . . . . .	45
<a href="#">Chart2D::ChartOptions</a>	It defines some features of the chart . . . . .	51
<a href="#">Compressor</a>	Audio compressor effect . . . . .	52
<a href="#">DecodingType</a>	SAC decoder parameter decoding type . . . . .	54
<a href="#">Effect</a>	<a href="#">Effect</a> class. It contains (by inheritance) all effects classes . . . . .	55
<a href="#">EffectBase</a>	<a href="#">Effect</a> base class . . . . .	66
<a href="#">EffectsMonitor</a>	Class for managing effects parameters . . . . .	69
<a href="#">Encoder</a>	<a href="#">Encoder</a> window interface . . . . .	76
<a href="#">File::Endianness</a>	. . . . .	81
<a href="#">Equalizer</a>	Audio equalizer effect . . . . .	81

<a href="#">File</a>	
Audio file class	86
<a href="#">WAVFile::Header</a>	
Audio file header struct	96
<a href="#">HRTFModel</a>	
SAC decoder parameter HRTF model	100
<a href="#">LogType</a>	101
<a href="#">OutputDevice</a>	
Audio output device class (QIODevice extension)	101
<a href="#">Panning</a>	
Audio panning effect	107
<a href="#">ProcessManager</a>	
Process manager class. It contains all functions to perform the signal treatment process	108
<a href="#">Reverb</a>	
Audio reverb effect	115
<a href="#">SACBitstream</a>	
SAC bitstream class	121
<a href="#">SACEffects</a>	
<a href="#">SACEffects</a> window interface	124
<a href="#">UpmixType</a>	
SAC decoder parameter upmix type	134
<a href="#">WAVFile</a>	
Audio file as WAV format class	135

## Chapter 4

# File Index

### 4.1 File List

Here is a list of all documented files with brief descriptions:

src/main.cpp	??
src/effects/Compressor.cpp	??
src/effects/Compressor.h	??
src/effects/Effect.cpp	??
src/effects/Effect.h	??
src/effects/EffectBase.h	??
src/effects/Equalizer.cpp	??
src/effects/Equalizer.h	??
src/effects/Panning.cpp	??
src/effects/Panning.h	??
src/effects/Reverb.cpp	??
src/effects/Reverb.h	??
src/interface/AudioInfo.cpp	??
src/interface/AudioInfo.h	??
src/interface/AudioOutput.cpp	??
src/interface/AudioOutput.h	??
src/interface/ChannelsList.cpp	??
src/interface/ChannelsList.h	??
src/interface/Chart2D.cpp	??
src/interface/Chart2D.h	??
src/interface/EffectsMonitor.cpp	??
src/interface/EffectsMonitor.h	??
src/interface/Encoder.cpp	??
src/interface/Encoder.h	??
src/interface/main.cpp	??
src/interface/SACEffects.cpp	??
src/interface/SACEffects.h	??
src/process/AudioSignal.cpp	??
src/process/AudioSignal.h	??
src/process/File.cpp	??
src/process/File.h	??
src/process/ProcessManager.cpp	??
src/process/ProcessManager.h	??
src/sac/sac_decoder.c	??
src/sac/sac_decoder.h	??

src/sac/ <b>sac_encoder.c</b> . . . . .	??
src/sac/ <b>sac_encoder.h</b> . . . . .	??
src/sac/ <b>SACBitstream.cpp</b> . . . . .	??
src/sac/ <b>SACBitstream.h</b> . . . . .	??
src/tools/ <a href="#">Logger.cpp</a>	
Functions to create log messages on console . . . . .	<a href="#">143</a>
src/tools/ <b>Logger.h</b> . . . . .	??

## Chapter 5

# Namespace Documentation

### 5.1 Ui Namespace Reference

Qt user interfaces namespace (see <http://doc.qt.io/qt-5/topics-ui.html>).





## Chapter 6

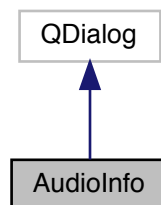
# Class Documentation

### 6.1 AudiolInfo Class Reference

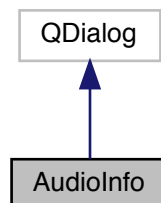
Audio object info dialog class.

```
#include <AudioInfo.h>
```

Inheritance diagram for AudiolInfo:



Collaboration diagram for AudiolInfo:



## Public Member Functions

- [AudioInfo](#) (QWidget \*parent=0)  
*AudioInfo constructor.*
- [~AudioInfo](#) ()  
*AudioInfo destructor.*
- void [setFile](#) (WAVFile \*file)  
*It sets a audio file.*

### 6.1.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 23 of file AudioInfo.h.

### 6.1.2 Constructor & Destructor Documentation

#### 6.1.2.1 AudioInfo()

```
AudioInfo::AudioInfo (
    QWidget * parent = 0 )
```

#### Parameters

<i>parent</i>	user interface parent object
---------------	------------------------------

Definition at line 8 of file AudioInfo.cpp.

Here is the call graph for this function:



### 6.1.3 Member Function Documentation

## 6.1.3.1 setFile()

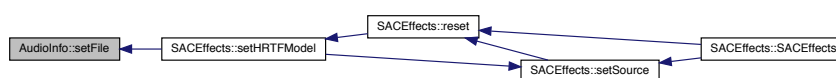
```
void AudioInfo::setFile (
    WAVFile * file )
```

## Parameters

<i>file</i>	audio file object
-------------	-------------------

Definition at line 26 of file AudioInfo.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

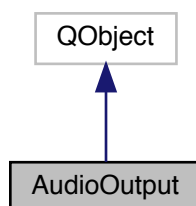
- `src/interface/AudioInfo.h`
- `src/interface/AudioInfo.cpp`

## 6.2 AudioOutput Class Reference

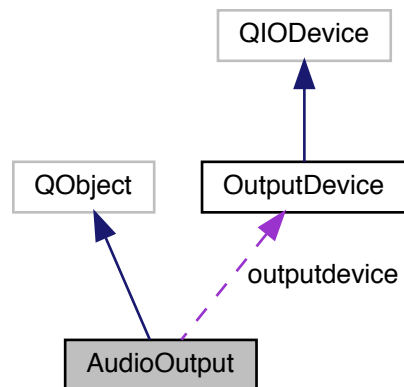
Audio output class.

```
#include <AudioOutput.h>
```

Inheritance diagram for AudioOutput:



Collaboration diagram for AudioOutput:



## Public Slots

### User interface slots

They are called when a user interface element is being changed.

- void [setDevice](#) (int index)  
*It selects an output device.*

## Public Member Functions

- [AudioOutput](#) (QComboBox \*selector, int [fs](#), int [samplesize](#))  
*AudioOutput constructor.*
- [~AudioOutput](#) ()  
*AudioOutput destructor.*
- void [start](#) ()  
*It resumes audio output playback.*
- void [stop](#) ()  
*It stops audio output playback.*
- void [setFormat](#) (int [fs](#), int [samplesize](#))  
*It sets signal sampling frequency.*
- void [setDevices](#) ()  
*It sets all available audio output devices.*
- void [setDevices](#) (QList< QAudioDeviceInfo > devices)  
*It sets a list of audio devices.*
- void [setVolume](#) (float [volume](#))  
*It sets audio output volume level.*

## Public Attributes

- [OutputDevice](#) \* `outputdevice`
- int `fs`
- int `samplesize`
- float `volume`

### 6.2.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 59 of file AudioOutput.h.

### 6.2.2 Constructor & Destructor Documentation

#### 6.2.2.1 AudioOutput()

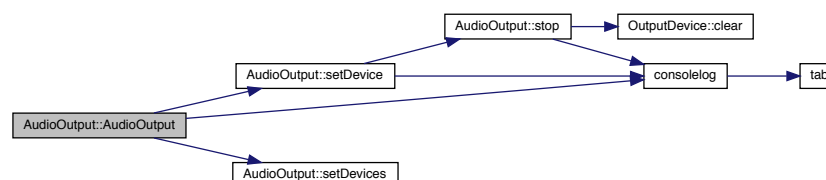
```
AudioOutput::AudioOutput (
    QComboBox * selector,
    int fs,
    int samplesize )
```

#### Parameters

<i>selector</i>	user interface combo box to select audio device
<i>fs</i>	signal sampling frequency
<i>samplesize</i>	audio sample size [bits]

Definition at line 10 of file AudioOutput.cpp.

Here is the call graph for this function:



### 6.2.3 Member Function Documentation

### 6.2.3.1 setDevice

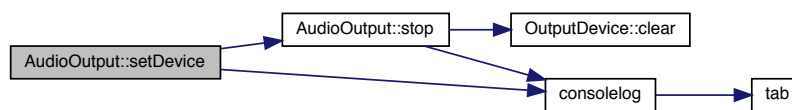
```
void AudioOutput::setDevice (
    int index ) [slot]
```

#### Parameters

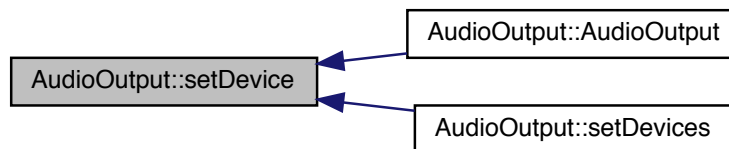
<i>index</i>	device index
--------------	--------------

Definition at line 137 of file AudioOutput.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.2.3.2 setDevices()

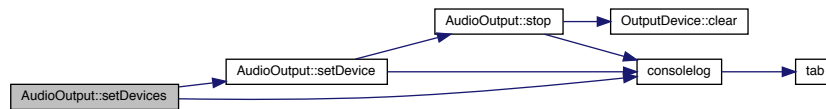
```
void AudioOutput::setDevices (
    QList< QAudioDeviceInfo > devices )
```

#### Parameters

<i>devices</i>	
----------------	--

Definition at line 100 of file AudioOutput.cpp.

Here is the call graph for this function:



### 6.2.3.3 setFormat()

```
void AudioOutput::setFormat (
    int fs,
    int samplesize )
```

#### Parameters

<i>fs</i>	signal sampling frequency.
<i>samplesize</i>	signal sample size

Definition at line 83 of file AudioOutput.cpp.

Here is the caller graph for this function:



### 6.2.3.4 setVolume()

```
void AudioOutput::setVolume (
    float volume )
```

#### Parameters

<i>volume</i>	real number from 0 to 1
---------------	-------------------------

Definition at line 122 of file AudioOutput.cpp.

## 6.2.4 Member Data Documentation

### 6.2.4.1 fs

```
int AudioOutput::fs
```

signal sampling frequency [Hz]

Definition at line 63 of file AudioOutput.h.

### 6.2.4.2 outputdevice

```
OutputDevice* AudioOutput::outputdevice
```

audio output QIODevice class object to control audio output device functions

Definition at line 62 of file AudioOutput.h.

### 6.2.4.3 samplesize

```
int AudioOutput::samplesize
```

audio sample size [bits]

Definition at line 64 of file AudioOutput.h.

### 6.2.4.4 volume

```
float AudioOutput::volume
```

audio output volume

Definition at line 65 of file AudioOutput.h.

The documentation for this class was generated from the following files:

- src/interface/AudioOutput.h
- src/interface/AudioOutput.cpp



## 6.3 AudioSignal Class Reference

Audio signal class.

```
#include <AudioSignal.h>
```

### Public Member Functions

- [AudioSignal](#) (int [fs](#))  
*AudioSignal* constructor (empty signal vector).
- [AudioSignal](#) (std::vector< float > signal, int [fs](#))  
*AudioSignal* constructor.
- [~AudioSignal](#) ()  
*AudioSignal* destructor.
- float [operator\[\]](#) (int index)  
*It gets a sample from the selected index.*
- [AudioSignal](#) [getSample](#) (int start, int end)  
*It gets samples from a specific range.*
- void [setSample](#) (int index, float sample)  
*It sets a sample in the selected index.*
- void [addSample](#) (float sample)  
*It adds a sample to the end of the signal.*
- void [deleteSample](#) (int index)  
*It deletes a sample at a selected position.*
- void [deleteSample](#) (int start, int end)  
*It deletes a range of samples.*
- std::vector< float > [getSignal](#) ()  
*It gets the entire signal.*
- void [setSignal](#) (std::vector< float > signal)  
*It sets the entire signal.*
- std::vector< float > [getTimes](#) ()  
*It gets time [s] axis as a vector beginning at time  $t = 0$  s.*
- std::vector< float > [getTimes](#) (float initialtime)  
*It gets time [s] axis as a vector beginning at a specific initial time.*
- std::vector< float > [getSpectrum](#) ()  
*It gets the signal spectral density.*
- std::vector< float > [getSpectrum](#) (int bands)  
*It gets the signal spectral density.*
- std::vector< float > [getFrequencies](#) ()  
*It gets frequencies [Hz] axis as a vector.*
- std::vector< float > [getFrequencies](#) (int bands)  
*It gets frequencies [Hz] axis as a vector.*
- void [clear](#) ()  
*It removes all samples from the signal.*

### Public Attributes

- int [size](#)
- int [fs](#)

## Static Public Attributes

- static const unsigned int `maxsamples` = 0xFFFFF

### 6.3.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 17 of file AudioSignal.h.

### 6.3.2 Constructor & Destructor Documentation

#### 6.3.2.1 AudioSignal() [1/2]

```
AudioSignal::AudioSignal (
    int fs )
```

#### Parameters

<i>fs</i>	signal sampling frequency [Hz]
-----------	--------------------------------

Definition at line 18 of file AudioSignal.cpp.

Here is the caller graph for this function:



#### 6.3.2.2 AudioSignal() [2/2]

```
AudioSignal::AudioSignal (
    std::vector< float > signal,
    int fs )
```

## Parameters

<i>signal</i>	vector of signal samples
<i>fs</i>	signal sampling frequency [Hz]

Definition at line 29 of file AudioSignal.cpp.

### 6.3.3 Member Function Documentation

#### 6.3.3.1 addSample()

```
void AudioSignal::addSample (  
    float sample )
```

## Parameters

<i>sample</i>	new signal sample
---------------	-------------------

Definition at line 89 of file AudioSignal.cpp.

Here is the call graph for this function:



#### 6.3.3.2 deleteSample() [1/2]

```
void AudioSignal::deleteSample (  
    int index )
```

## Parameters

<i>index</i>	sample position index
--------------	-----------------------

Definition at line 104 of file AudioSignal.cpp.

**6.3.3.3 deleteSample()** [2/2]

```
void AudioSignal::deleteSample (
    int start,
    int end )
```

**Parameters**

<i>start</i>	first index of the range (included)
<i>end</i>	last index of the range (included)

Definition at line 114 of file AudioSignal.cpp.

**6.3.3.4 getFrequencies()** [1/2]

```
std::vector< float > AudioSignal::getFrequencies ( )
```

**Returns**

frequencies vector

Definition at line 227 of file AudioSignal.cpp.

Here is the caller graph for this function:

**6.3.3.5 getFrequencies()** [2/2]

```
std::vector< float > AudioSignal::getFrequencies (
    int bands )
```

**Parameters**

<i>bands</i>	number of frequency bands of the signal spectral density (if higher number than available has been requested, it returns as the highest number of frequency as possible)
--------------	--

**Returns**

frequencies vector

Definition at line 241 of file AudioSignal.cpp.

Here is the call graph for this function:

**6.3.3.6 getSample()**

```
AudioSignal AudioSignal::getSample (
    int start,
    int end )
```

**Parameters**

<i>start</i>	first index of the range (included)
<i>end</i>	last index of the range (included)

**Returns**

subsignal object

Definition at line 64 of file AudioSignal.cpp.

Here is the call graph for this function:



### 6.3.3.7 `getSignal()`

```
std::vector< float > AudioSignal::getSignal ( )
```

#### Returns

audio signal

Definition at line 124 of file AudioSignal.cpp.

### 6.3.3.8 `getSpectrum()` [1/2]

```
std::vector< float > AudioSignal::getSpectrum ( )
```

#### Returns

signal spectral density

Definition at line 169 of file AudioSignal.cpp.

Here is the caller graph for this function:



### 6.3.3.9 `getSpectrum()` [2/2]

```
std::vector< float > AudioSignal::getSpectrum (
    int bands )
```

#### Parameters

<i>bands</i>	number of frequency bands of the signal spectral density (if higher number than available has been requested, it returns as the highest number of frequency as possible)
--------------	--

#### Returns

signal spectral density

Definition at line 195 of file AudioSignal.cpp.

Here is the call graph for this function:



#### 6.3.3.10 `getTimes()` [1/2]

```
std::vector< float > AudioSignal::getTimes ( )
```

##### Returns

time vector

Definition at line 141 of file AudioSignal.cpp.

#### 6.3.3.11 `getTimes()` [2/2]

```
std::vector< float > AudioSignal::getTimes (
    float initialtime )
```

##### Parameters

<i>initialtime</i>	initial time [s]
--------------------	------------------

##### Returns

time vector

Definition at line 155 of file AudioSignal.cpp.

#### 6.3.3.12 `operator[]()`

```
float AudioSignal::operator[] (
    int index )
```

**Parameters**

<i>index</i>	selected index
--------------	----------------

**Returns**

sample signal sample

Definition at line 46 of file AudioSignal.cpp.

Here is the call graph for this function:

**6.3.3.13 setSample()**

```
void AudioSignal::setSample (
    int index,
    float sample )
```

**Parameters**

<i>index</i>	selected index
<i>sample</i>	new signal sample

Definition at line 76 of file AudioSignal.cpp.

Here is the call graph for this function:





#### 6.3.3.14 setSignal()

```
void AudioSignal::setSignal (
    std::vector< float > signal )
```

##### Parameters

<i>signal</i>	audio signal
---------------	--------------

Definition at line 132 of file AudioSignal.cpp.

### 6.3.4 Member Data Documentation

#### 6.3.4.1 fs

```
int AudioSignal::fs
```

signal sampling frequency [Hz]

Definition at line 20 of file AudioSignal.h.

#### 6.3.4.2 maxsamples

```
const unsigned int AudioSignal::maxsamples = 0xFFFFF [static]
```

maximum number of samples

Definition at line 21 of file AudioSignal.h.

#### 6.3.4.3 size

```
int AudioSignal::size
```

number of samples

Definition at line 19 of file AudioSignal.h.

The documentation for this class was generated from the following files:

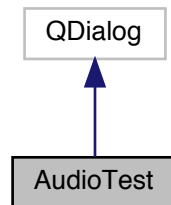
- src/process/AudioSignal.h
- src/process/AudioSignal.cpp

## 6.4 AudioTest Class Reference

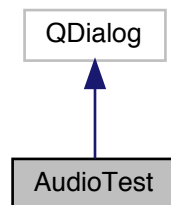
Audio output test class.

```
#include <AudioOutput.h>
```

Inheritance diagram for AudioTest:



Collaboration diagram for AudioTest:



### Public Member Functions

- [AudioTest](#) (QWidget \*parent=0)  
*AudioTest constructor.*
- [~AudioTest](#) ()  
*AudioTest destructor.*

#### 6.4.1 Detailed Description

Author

Andrés González Fornell

Definition at line 89 of file AudioOutput.h.

## 6.4.2 Constructor & Destructor Documentation

### 6.4.2.1 AudioTest()

```
AudioTest::AudioTest (
    QWidget * parent = 0 )
```

#### Parameters

<i>parent</i>	window parent
---------------	---------------

Definition at line 298 of file AudioOutput.cpp.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- src/interface/AudioOutput.h
- src/interface/AudioOutput.cpp

## 6.5 BinauralQuality Struct Reference

SAC decoder parameter binaural quality.

```
#include <SACEffects.h>
```

### Public Types

- enum [binauralquality](#) { **parametric** = 0, [filtering](#) = 1 }

### 6.5.1 Detailed Description

Definition at line 46 of file SACEffects.h.

## 6.5.2 Member Enumeration Documentation

### 6.5.2.1 binauralquality

enum `BinauralQuality::binauralquality`

## Enumerator

filtering	parametric binaural quality filtering binaural quality
-----------	--

Definition at line 47 of file SACEffects.h.

The documentation for this struct was generated from the following file:

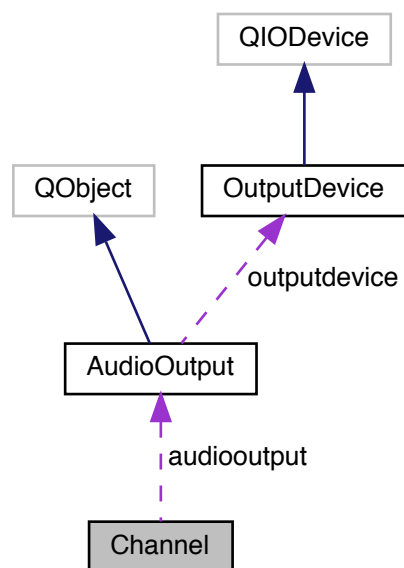
- src/interface/SACEffects.h

## 6.6 Channel Class Reference

Single-object class from channels list.

```
#include <ChannelsList.h>
```

Collaboration diagram for Channel:



### Public Member Functions

- [Channel](#) (QLayout \*framework, std::string prefix, int [index](#), bool isoutput)  
*Channels constructor.*
- [~Channel](#) ()  
*Channels desctructor.*
- int [getIndex](#) ()

*It gets the channel index.*

- void **setIndex** (int [index](#))
- void **setLabel** (std::string [label](#))

*It sets a label to the channel name, i.e., group box title and label text.*

- void **setVolume** (int [volume](#))

*It sets the channel volume level.*

- void **mute** (bool state)

*It mutes channel.*

- void **bypass** (bool state)

*It sets channel to bypass effects.*

## Public Attributes

- int [index](#)
- std::string [name](#)
- double [volume](#)
- bool [muted](#)
- bool [bypassed](#)
- [AudioOutput](#) \* [audiooutput](#)

- QGroupBox \* [groupbox](#)

*user interface elements*

- QLineEdit \* [label](#)
- QSlider \* [volumeslider](#)
- QCheckBox \* [mutecheckbox](#)
- QCheckBox \* [bypasscheckbox](#)
- QComboBox \* [deviceselector](#)

## 6.6.1 Detailed Description

### Author

Andrés González Fornell

Definition at line 32 of file ChannelsList.h.

## 6.6.2 Constructor & Destructor Documentation

### 6.6.2.1 Channel()

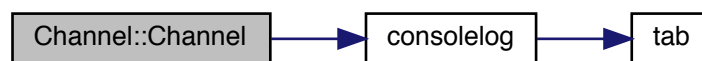
```
Channel::Channel (
    QLayout * framework,
    std::string prefix,
    int index,
    bool isoutput )
```

## Parameters

<i>framework</i>	channel user interface framework
<i>prefix</i>	prefix of objects name of channel user interface
<i>index</i>	channel index
<i>isoutput</i>	true to create device selector to send audio to the system audio output devices

Definition at line 213 of file ChannelsList.cpp.

Here is the call graph for this function:



### 6.6.3 Member Function Documentation

#### 6.6.3.1 `bypass()`

```
void Channel::bypass (
    bool state )
```

## Parameters

<i>state</i>	true to bypass effects and false to apply them
--------------	--

Definition at line 318 of file ChannelsList.cpp.

#### 6.6.3.2 `getIndex()`

```
int Channel::getIndex ( )
```

## Returns

index

Definition at line 271 of file ChannelsList.cpp.

#### 6.6.3.3 mute()

```
void Channel::mute (
    bool state )
```

##### Parameters

<i>state</i>	true to mute the channel and false to unmuted
--------------	---

Definition at line 309 of file ChannelsList.cpp.

#### 6.6.3.4 setLabel()

```
void Channel::setLabel (
    std::string label )
```

##### Parameters

<i>label</i>	new label
--------------	-----------

Definition at line 297 of file ChannelsList.cpp.

#### 6.6.3.5 setVolume()

```
void Channel::setVolume (
    int volume )
```

##### Parameters

<i>volume</i>	integer number from 0 to 100
---------------	------------------------------

Definition at line 327 of file ChannelsList.cpp.

### 6.6.4 Member Data Documentation

#### 6.6.4.1 audiooutput

[AudioOutput\\*](#) Channel::audiooutput

system audio output devices object

Definition at line 39 of file ChannelsList.h.



#### 6.6.4.2 bypasscheckbox

```
QCheckBox* Channel::bypasscheckbox
```

checkbox object to bypass effect

Definition at line 48 of file ChannelsList.h.

#### 6.6.4.3 bypassed

```
bool Channel::bypassed
```

it tells channel to bypass effects or apply them

Definition at line 38 of file ChannelsList.h.

#### 6.6.4.4 deviceselector

```
QComboBox* Channel::deviceselector
```

audio output device selector object

Definition at line 49 of file ChannelsList.h.

#### 6.6.4.5 groupbox

```
QGroupBox* Channel::groupbox
```

channel group box

Definition at line 44 of file ChannelsList.h.

#### 6.6.4.6 index

```
int Channel::index
```

channel index

Definition at line 34 of file ChannelsList.h.

#### 6.6.4.7 label

```
QLineEdit* Channel::label
```

field to change the channel label

Definition at line 45 of file ChannelsList.h.

#### 6.6.4.8 mutecheckbox

```
QCheckBox* Channel::mutecheckbox
```

muted checkbox object

Definition at line 47 of file ChannelsList.h.

#### 6.6.4.9 muted

```
bool Channel::muted
```

it indicates if channel is muted

Definition at line 37 of file ChannelsList.h.

#### 6.6.4.10 name

```
std::string Channel::name
```

channel name

Definition at line 35 of file ChannelsList.h.

#### 6.6.4.11 volume

```
double Channel::volume
```

current audio volume level

Definition at line 36 of file ChannelsList.h.

#### 6.6.4.12 volumeslider

`QSlider* Channel::volumeslider`

volume level slider

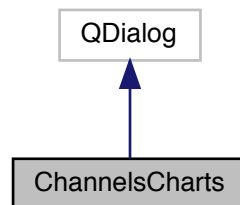
Definition at line 46 of file ChannelsList.h.

The documentation for this class was generated from the following files:

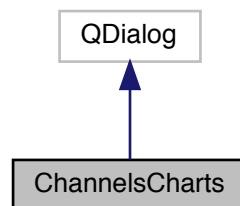
- `src/interface/ChannelsList.h`
- `src/interface/ChannelsList.cpp`

## 6.7 ChannelsCharts Class Reference

Inheritance diagram for ChannelsCharts:



Collaboration diagram for ChannelsCharts:



### Public Member Functions

- [ChannelsCharts](#) (float \*\*input, float \*\*output, [ChannelsList](#) \*input\_channels, [ChannelsList](#) \*output\_channels, int samples, QWidget \*parent=0)  
*[ChannelsCharts](#) constructor.*
- [~ChannelsCharts](#) ()  
*[ChannelsCharts](#) destructor.*

### 6.7.1 Detailed Description

Definition at line 103 of file ChannelsList.h.

### 6.7.2 Constructor & Destructor Documentation

#### 6.7.2.1 ChannelsCharts()

```
ChannelsCharts::ChannelsCharts (
    float ** input,
    float ** output,
    ChannelsList * input_channels,
    ChannelsList * output_channels,
    int samples,
    QWidget * parent = 0 )
```

##### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>input_channels</i>	input channels object
<i>output_channels</i>	output channels object
<i>samples</i>	number of samples each channel
<i>parent</i>	user interface parent object

Definition at line 344 of file ChannelsList.cpp.

The documentation for this class was generated from the following files:

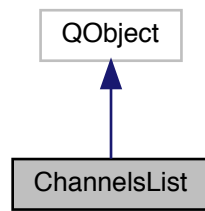
- src/interface/ChannelsList.h
- src/interface/ChannelsList.cpp

## 6.8 ChannelsList Class Reference

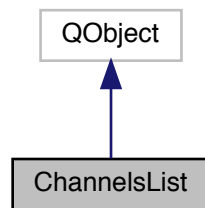
Channels list class. It shows information about channels signals.

```
#include <ChannelsList.h>
```

Inheritance diagram for ChannelsList:



Collaboration diagram for ChannelsList:



## Signals

- void **namechanged** (QString, int)

## Public Member Functions

- [ChannelsList](#) (QWidget \*framework, int number, bool showdevices)  
*ChannelsList constructor.*
- [~ChannelsList](#) ()  
*ChannelsList destructor.*
- [Channel \\*](#) [getChannel](#) (int index)  
*It gets a channel.*
- void [deleteChannel](#) (int index)  
*It deletes a channel.*
- int [getSize](#) ()  
*It gets the number of channels.*
- void [setSize](#) (int size)  
*It sets a number of channels up.*
- [std::vector< std::string >](#) [getNames](#) ()  
*It gets all channels names.*

## Static Public Attributes

- static int [fs](#)
- static int [samplesize](#)

### 6.8.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 69 of file ChannelsList.h.

### 6.8.2 Constructor & Destructor Documentation

#### 6.8.2.1 ChannelsList()

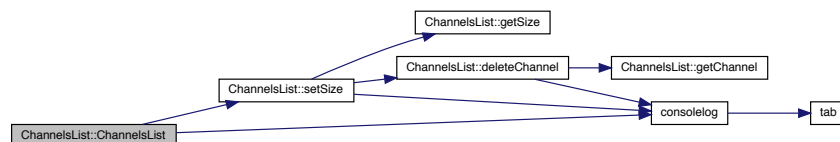
```
ChannelsList::ChannelsList (
    QWidget * framework,
    int number,
    bool showdevices )
```

#### Parameters

<i>framework</i>	user interface framework of channels list
<i>number</i>	number of channels
<i>showdevices</i>	true to create device selector to send audio to the system audio output devices

Definition at line 10 of file ChannelsList.cpp.

Here is the call graph for this function:



### 6.8.3 Member Function Documentation

## 6.8.3.1 deleteChannel()

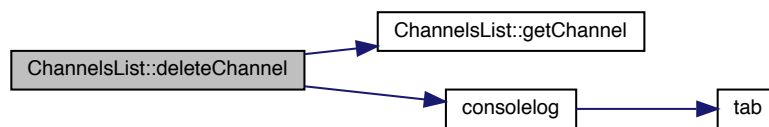
```
void ChannelsList::deleteChannel (
    int index )
```

## Parameters

<i>index</i>	channel index
--------------	---------------

Definition at line 40 of file ChannelsList.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.8.3.2 getChannel()

```
Channel * ChannelsList::getChannel (
    int index )
```

## Parameters

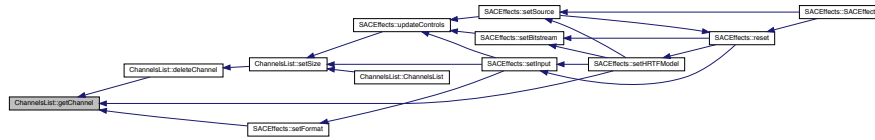
<i>index</i>	channel index
--------------	---------------

## Returns

channel pointer

Definition at line 32 of file ChannelsList.cpp.

Here is the caller graph for this function:



### 6.8.3.3 getNames()

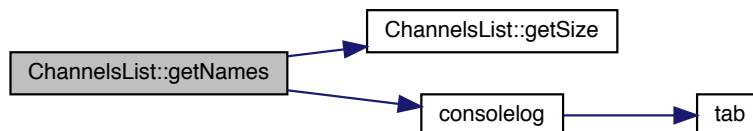
```
std::vector< std::string > ChannelsList::getNames ( )
```

#### Returns

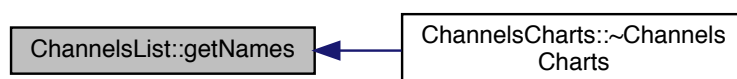
list of channels names

Definition at line 100 of file ChannelsList.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:





## 6.8.3.4 getSize()

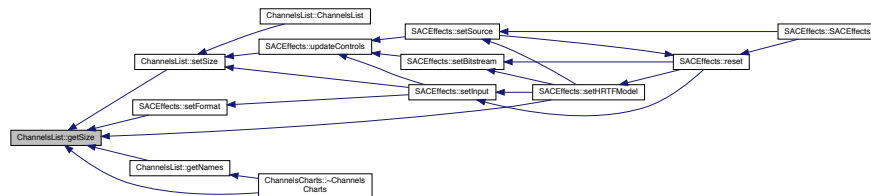
```
int ChannelsList::getSize ( )
```

## Returns

number of channels

Definition at line 53 of file ChannelsList.cpp.

Here is the caller graph for this function:



## 6.8.3.5 setSize()

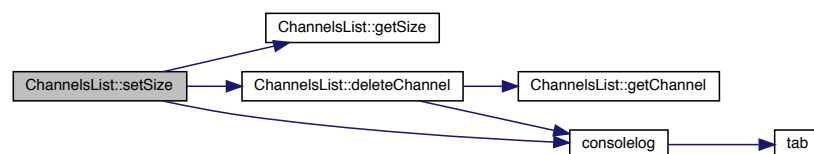
```
void ChannelsList::setSize (
    int size )
```

## Parameters

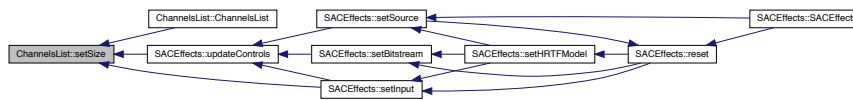
<i>size</i>	number of channels
-------------	--------------------

Definition at line 61 of file ChannelsList.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.8.4 Member Data Documentation

### 6.8.4.1 fs

```
int ChannelsList::fs [static]
```

signal sampling frequency

Definition at line 72 of file ChannelsList.h.

### 6.8.4.2 samplesize

```
int ChannelsList::samplesize [static]
```

signal sample size

Definition at line 73 of file ChannelsList.h.

The documentation for this class was generated from the following files:

- src/interface/ChannelsList.h
- src/interface/ChannelsList.cpp
- src/interface/SACEffects.cpp

## 6.9 SACBitstream::ChannelType Struct Reference

It specifies the channel type.

```
#include <SACBitstream.h>
```

### Public Types

- enum [channeltype](#) {  
L = 0x0, Lc = 0x1, Ls = 0x2, Lsr = 0x3,  
R = 0x4, Rc = 0x5, Rs = 0x6, Rsr = 0x7,  
C = 0x8, LFE = 0x9 }

### 6.9.1 Detailed Description

Definition at line 21 of file SACBitstream.h.

### 6.9.2 Member Enumeration Documentation

#### 6.9.2.1 channeltype

```
enum SACBitstream::ChannelType::channeltype
```

##### Enumerator

L	left front channel
Lc	left front center channel
Ls	left surround channel
Lsr	rear surround left channel
R	left front channel
Rc	left front center channel
Rs	left surround channel
Rsr	rear surround left channel
C	center front channel
LFE	low frequency enhancement channel

Definition at line 22 of file SACBitstream.h.

The documentation for this struct was generated from the following file:

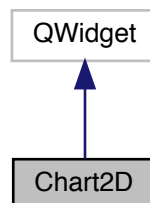
- src/sac/SACBitstream.h

## 6.10 Chart2D Class Reference

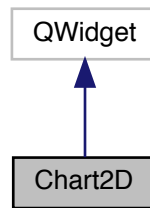
Class for plotting two-dimensional charts.

```
#include <Chart2D.h>
```

Inheritance diagram for Chart2D:



Collaboration diagram for Chart2D:



## Classes

- struct [ChartOptions](#)  
*It defines some features of the chart.*

## Public Member Functions

- [Chart2D](#) (QWidget \*framework)  
*Chart constructor.*
- [Chart2D](#) (QWidget \*framework, double range[2][2], std::string title, std::string [xlabel](#), std::string [ylabel](#), int options)  
*Chart constructor.*
- [~Chart2D](#) ()  
*Chart destructor.*
- void [setPoints](#) (QVector< QPointF > points)  
*It sets the points to the chart serie.*
- QVector< QPointF > [getPoints](#) ()  
*It gets the points from the chart serie.*
- void [setRange](#) (double range[2][2])  
*It sets the axis range.*
- void [setTitle](#) (std::string title)  
*It sets chart title.*
- void [setOptions](#) (int options)  
*It sets chart options.*
- void [clear](#) ()  
*It clears the chart.*

## Public Attributes

- std::string [xlabel](#)
- std::string [ylabel](#)

### 6.10.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 22 of file Chart2D.h.

### 6.10.2 Constructor & Destructor Documentation

#### 6.10.2.1 Chart2D() [1/2]

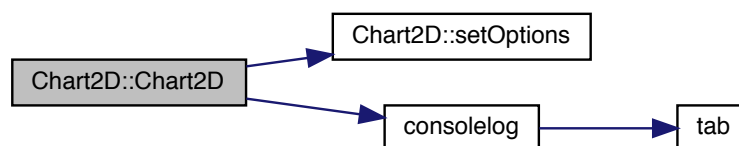
```
Chart2D::Chart2D (
    QWidget * framework )
```

#### Parameters

<i>framework</i>	user interface framework of chart
------------------	-----------------------------------

Definition at line 8 of file Chart2D.cpp.

Here is the call graph for this function:



#### 6.10.2.2 Chart2D() [2/2]

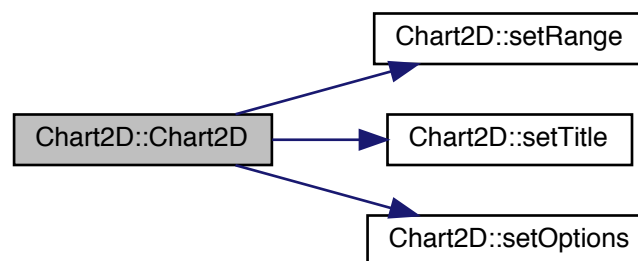
```
Chart2D::Chart2D (
    QWidget * framework,
    double range[2][2],
    std::string title,
    std::string xlabel,
    std::string ylabel,
    int options )
```

## Parameters

<i>framework</i>	user interface framework of chart
<i>range</i>	axes range matrix (range[0][0] = x_min, range[0][1] = x_max, range[1][0] = y_min, range[1][1] = y_max)
<i>title</i>	chart title (it will be impress on the chart)
<i>xlabel</i>	label for horizontal (x) axis
<i>ylabel</i>	label for vertical (y) axis
<i>options</i>	<a href="#">ChartOptions</a>

Definition at line 31 of file Chart2D.cpp.

Here is the call graph for this function:



### 6.10.3 Member Function Documentation

#### 6.10.3.1 `getPoints()`

```
QVector< QPointF > Chart2D::getPoints ( )
```

##### Returns

points chart points

Definition at line 60 of file Chart2D.cpp.

#### 6.10.3.2 `setOptions()`

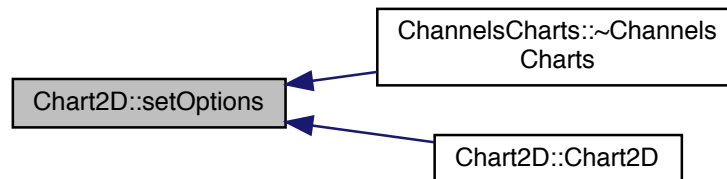
```
void Chart2D::setOptions (
    int options )
```

## Parameters

<i>options</i>	<a href="#">ChartOptions</a>
----------------	------------------------------

Definition at line 86 of file Chart2D.cpp.

Here is the caller graph for this function:



### 6.10.3.3 setPoints()

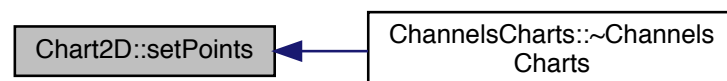
```
void Chart2D::setPoints (
    QVector< QPointF > points )
```

## Parameters

<i>points</i>	new chart points
---------------	------------------

Definition at line 52 of file Chart2D.cpp.

Here is the caller graph for this function:



#### 6.10.3.4 setRange()

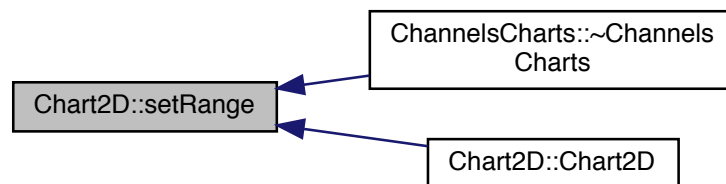
```
void Chart2D::setRange (
    double range[2][2] )
```

##### Parameters

<i>range</i>	axis range matrix (range[0][0] = x_min, range[0][1] = x_max, range[1][0] = y_min, range[1][1] = y_max)
--------------	--

Definition at line 68 of file Chart2D.cpp.

Here is the caller graph for this function:



#### 6.10.3.5 setTitle()

```
void Chart2D::setTitle (
    std::string title )
```

##### Parameters

<i>title</i>	chart title
--------------	-------------

Definition at line 77 of file Chart2D.cpp.

Here is the caller graph for this function:





## 6.10.4 Member Data Documentation

### 6.10.4.1 xlabel

```
std::string Chart2D::xlabel
```

horizontal (x) axis label

Definition at line 37 of file Chart2D.h.

### 6.10.4.2 ylabel

```
std::string Chart2D::ylabel
```

vertical (y) axis label

Definition at line 38 of file Chart2D.h.

The documentation for this class was generated from the following files:

- src/interface/Chart2D.h
- src/interface/Chart2D.cpp

## 6.11 Chart2D::ChartOptions Struct Reference

It defines some features of the chart.

```
#include <Chart2D.h>
```

### Public Types

- enum [Options](#) {  
    [logX](#) = 0x00001, [logY](#) = 0x00010, [labelX](#) = 0x00100, [labelY](#) = 0x01000,  
    [legend](#) = 0x10000 }

### 6.11.1 Detailed Description

Definition at line 28 of file Chart2D.h.

## 6.11.2 Member Enumeration Documentation

### 6.11.2.1 Options

```
enum Chart2D::ChartOptions::Options
```

**Enumerator**

logX	it configures the x axis as logarithm scale
logY	it configures the y axis as logarithm scale
labelX	it shows the x axis description on the chart
labelY	it shows the y axis description on the chart
legend	it shows the legend on the chart

Definition at line 29 of file Chart2D.h.

The documentation for this struct was generated from the following file:

- src/interface/Chart2D.h

## 6.12 Compressor Class Reference

Audio compressor effect.

```
#include <Compressor.h>
```

### Public Member Functions

- [Compressor](#) ()  
*Compressor constructor.*
- void [apply](#) (float \*\*input, float \*\*output, int samples, std::vector< [SACBitstream::ChannelType::channeltype](#) > channels)  
*It applies compression effect.*
- std::vector< std::vector< double > > [plot](#) (std::string chart)  
*It sends some values to user interface charts.*
- void [update](#) ()  
*It sets params from map of params.*

### 6.12.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 12 of file Compressor.h.

### 6.12.2 Member Function Documentation

#### 6.12.2.1 [apply\(\)](#)

```
void Compressor::apply (
    float ** input,
    float ** output,
    int samples,
    std::vector< SACBitstream::ChannelType::channeltype > channels )
```

## Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>channels</i>	vector of channel types

Definition at line 16 of file Compressor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.12.2.2 plot()

```
std::vector< std::vector< double > > Compressor::plot (
    std::string chart )
```

## Parameters

<i>chart</i>	chart id
--------------	----------

## Returns

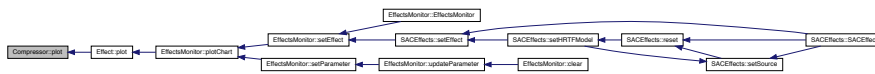
array of values as values[axis][sample] axis: 0 = x (horizontal) and 1 = y (vertical)

Definition at line 38 of file Compressor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- `src/effects/Compressor.h`
- `src/effects/Compressor.cpp`

## 6.13 DecodingType Struct Reference

SAC decoder parameter decoding type.

```
#include <SACEffects.h>
```

### Public Types

- enum `decodingtype` { `low` = 0, `high` = 1 }

#### 6.13.1 Detailed Description

Definition at line 37 of file `SACEffects.h`.

#### 6.13.2 Member Enumeration Documentation

##### 6.13.2.1 decodingtype

```
enum DecodingType::decodingtype
```

## Enumerator

low	low complexity decoding mode
high	high complexity decoding mode

Definition at line 38 of file SACEffects.h.

The documentation for this struct was generated from the following file:

- src/interface/SACEffects.h

## 6.14 Effect Class Reference

[Effect](#) class. It contains (by inheritance) all effects classes.

```
#include <Effect.h>
```

### Public Types

- enum [effectID](#) { [LIST](#) }  
*available effects enumeration*

### Public Member Functions

- [Effect](#) ([Effect::effectID](#) effect, int fs)  
*Effect constructor.*
- [Effect](#) ([Effect::effectID](#) effect, std::map< std::string, std::string > params, int fs)  
*Effect constructor.*
- [~Effect](#) ()  
*Effect destructor.*
- void [setParams](#) (std::map< std::string, std::string > params)  
*It sets parameters variable.*
- bool [apply](#) (float \*\*input, float \*\*output, int samples, std::vector< [SACBitstream::ChannelType::channeltype](#) > channels)  
*It applies the selected effect to the input and sets the result into output variable.*
- std::vector< std::vector< double > > [plot](#) (std::string chart)  
*It sends some values to user interface charts.*

### Static Public Member Functions

- static std::map< [Effect::effectID](#), std::string > [getEffects](#) ()  
*It gets the list of available effects.*
- static [Effect::effectID](#) [getEffect](#) (std::string effectname)  
*It gets effects type from the effect name.*
- static std::map< std::string, std::string > [getParams](#) (std::string configuration)  
*It gets params from a effect configuration file (.fx) text.*
- static std::vector< bool > [getChannels](#) (std::string configuration, int size)  
*It gets channels vector from a effect configuration file (.fx) text.*
- static std::vector< double > [getLevels](#) (std::string configuration, int size)  
*It gets levels vector from a effect configuration file (.fx) text.*
- static std::string [getTag](#) (std::string configuration, std::string tag)  
*It extracts the value in a tag from a effect configuration file (.fx) text.*
- static std::map< std::string, std::string > [getTagMap](#) (std::string configuration, std::string tag)  
*It extracts the map of values in a map-structured tag from a effect configuration file (.fx) text.*

## Public Attributes

- `std::pair< Effect::effectID, std::string > effect`

### 6.14.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 44 of file `Effect.h`.

### 6.14.2 Member Enumeration Documentation

#### 6.14.2.1 `effectID`

```
enum Effect::effectID
```

#### Enumerator

LIST	macros variable which contains all the effects
------	--

Definition at line 50 of file `Effect.h`.

### 6.14.3 Constructor & Destructor Documentation

#### 6.14.3.1 `Effect()` [1/2]

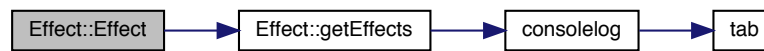
```
Effect::Effect (  
    Effect::effectID effect,  
    int fs )
```

#### Parameters

<i>effect</i>	effect ID
<i>fs</i>	signal sampling frequency

Definition at line 12 of file `Effect.cpp`.

Here is the call graph for this function:



### 6.14.3.2 Effect() [2/2]

```

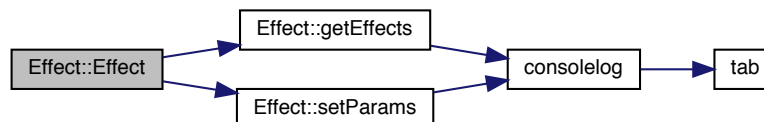
Effect::Effect (
    Effect::effectID effect,
    std::map< std::string, std::string > params,
    int fs )
  
```

#### Parameters

<i>effect</i>	effect ID
<i>params</i>	map of effect parameters
<i>fs</i>	signal sampling frequency

Definition at line 24 of file Effect.cpp.

Here is the call graph for this function:



## 6.14.4 Member Function Documentation

### 6.14.4.1 apply()

```

bool Effect::apply (
    float ** input,
    float ** output,
    int samples,
    std::vector< SACBitstream::ChannelType::channeltype > channels )
  
```

## Parameters

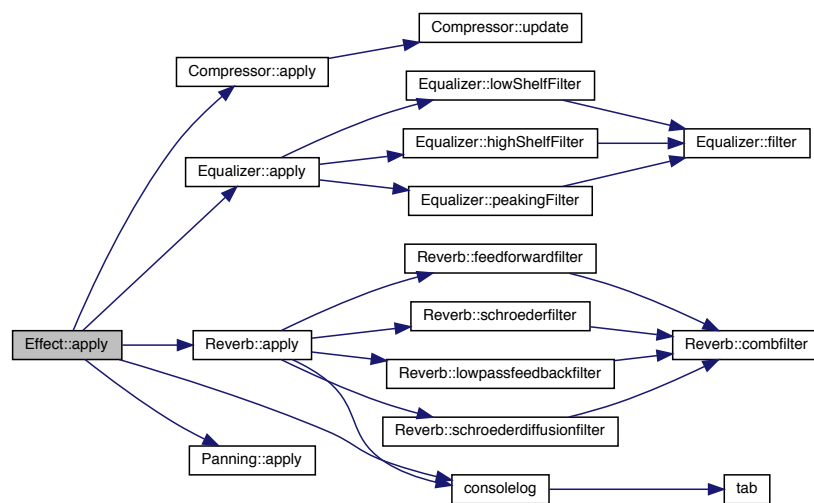
<i>input</i>	input data pointer
<i>output</i>	output data pointer
<i>samples</i>	number of samples
<i>channels</i>	vector of channel types

## Returns

true if it was successful

Definition at line 70 of file Effect.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.14.4.2 getChannels()

```

std::vector< bool > Effect::getChannels (
    std::string configuration,
    int size ) [static]
  
```



## Parameters

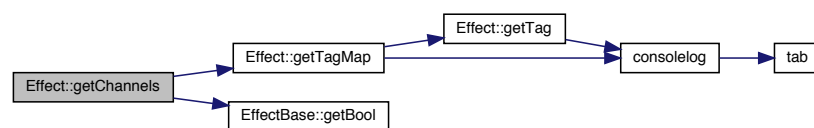
<i>configuration</i>	contained text of a effect configuration file (.fx)
<i>size</i>	number of channels

## Returns

channels boolean vector to select channels when applying effects

Definition at line 162 of file Effect.cpp.

Here is the call graph for this function:

6.14.4.3 `getEffect()`

```
Effect::effectID Effect::getEffect (
    std::string effectname ) [static]
```

## Parameters

<i>effectname</i>	effect name string
-------------------	--------------------

## Returns

effect type `effectID`

Definition at line 141 of file Effect.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.14.4.4 getEffects()

```
std::map< Effect::effectID, std::string > Effect::getEffects ( ) [static]
```

##### Returns

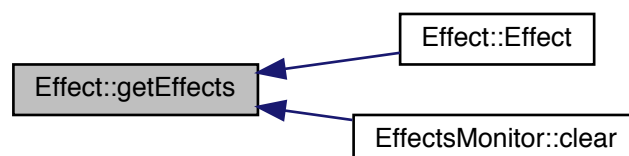
map of available effects

Definition at line 115 of file `Effect.cpp`.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.14.4.5 getLevels()

```
std::vector< double > Effect::getLevels (
    std::string configuration,
    int size ) [static]
```

**Parameters**

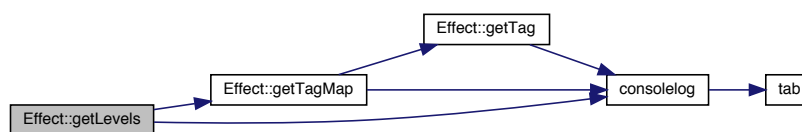
<i>configuration</i>	contained text of a effect configuration file (.fx)
<i>size</i>	number of channels

**Returns**

levels vector of input channels before applying effects

Definition at line 178 of file Effect.cpp.

Here is the call graph for this function:

**6.14.4.6 getParams()**

```
std::map< std::string, std::string > Effect::getParams (
    std::string configuration ) [static]
```

**Parameters**

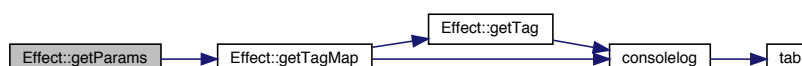
<i>configuration</i>	contained text of a effect configuration file (.fx)
----------------------	---

**Returns**

parameters map variable valid to apply effects

Definition at line 200 of file Effect.cpp.

Here is the call graph for this function:



## 6.14.4.7 getTag()

```
std::string Effect::getTag (
    std::string configuration,
    std::string tag ) [static]
```

## Parameters

<i>configuration</i>	contained text of a effect configuration file (.fx)
<i>tag</i>	tag name of the requested field

## Returns

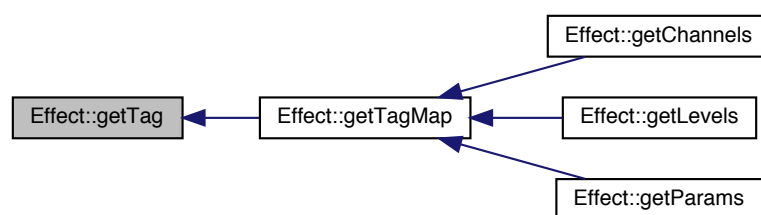
contained value in the tag

Definition at line 211 of file Effect.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.14.4.8 getTagMap()

```
std::map< std::string, std::string > Effect::getTagMap (
    std::string configuration,
    std::string tag ) [static]
```

## Parameters

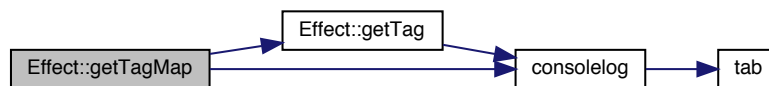
<i>configuration</i>	contained text of a effect configuration file (.fx)
<i>tag</i>	tag name of the requested field

## Returns

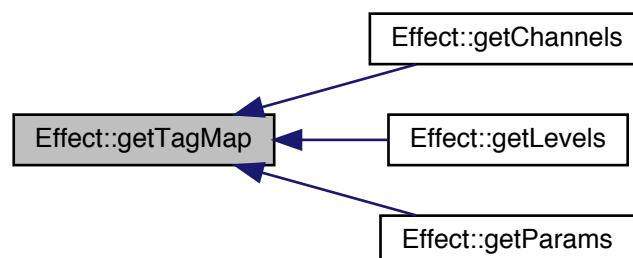
contained map of values in the tag

Definition at line 224 of file Effect.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.14.4.9 plot()

```
std::vector< std::vector< double > > Effect::plot (
    std::string chart )
```

## Parameters

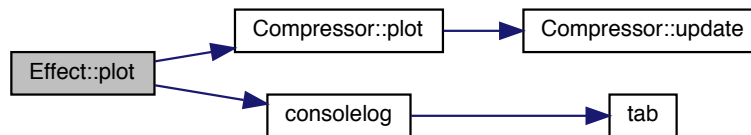
<i>chart</i>	chart id
--------------	----------

**Returns**

array of values as values[axis][sample] axis: 0 = x (horizontal) and 1 = y (vertical)

Definition at line 100 of file Effect.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.14.4.10 setParams()**

```
void Effect::setParams (
    std::map< std::string, std::string > params )
```

**Parameters**

<i>params</i>	parameters variable
---------------	---------------------

Definition at line 40 of file Effect.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.14.5 Member Data Documentation

### 6.14.5.1 effect

```
std::pair<Effect::effectID, std::string> Effect::effect
```

selected effect name and id

Definition at line 53 of file Effect.h.

The documentation for this class was generated from the following files:

- src/effects/Effect.h
- src/effects/Effect.cpp

## 6.15 EffectBase Class Reference

[Effect](#) base class.

```
#include <EffectBase.h>
```

### Public Member Functions

- [EffectBase](#) ()  
*EffectBase constructor.*

### Static Public Member Functions

- static int [getInt](#) (std::string param)  
*It parses a parameter value to double.*
- static double [getDouble](#) (std::string param)  
*It parses a parameter value to integer.*
- static bool [getBool](#) (std::string param)  
*It parses a parameter value to bool.*



## Static Public Attributes

- static int [fs](#)
- static std::map< std::string, std::string > [params](#)

### 6.15.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 21 of file EffectBase.h.

### 6.15.2 Member Function Documentation

#### 6.15.2.1 getBool()

```
bool EffectBase::getBool (
    std::string param ) [static]
```

#### Parameters

<i>param</i>	parameter value
--------------	-----------------

#### Returns

boolean value (false by default)

Definition at line 302 of file Effect.cpp.

Here is the caller graph for this function:



#### 6.15.2.2 getDouble()

```
double EffectBase::getDouble (
    std::string param ) [static]
```

**Parameters**

<i>param</i>	parameter value
--------------	-----------------

**Returns**

value

Definition at line 293 of file Effect.cpp.

**6.15.2.3 getInt()**

```
int EffectBase::getInt (
    std::string param ) [static]
```

**Parameters**

<i>param</i>	parameter value
--------------	-----------------

**Returns**

value as integer

Definition at line 284 of file Effect.cpp.

**6.15.3 Member Data Documentation****6.15.3.1 fs**

```
int EffectBase::fs [static]
```

signal sampling frequency [Hz]

Definition at line 23 of file EffectBase.h.

**6.15.3.2 params**

```
std::map< std::string, std::string > EffectBase::params [static]
```

string of effect parameters

Definition at line 24 of file EffectBase.h.

The documentation for this class was generated from the following files:

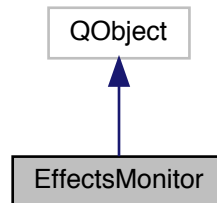
- src/effects/EffectBase.h
- src/effects/Effect.cpp

## 6.16 EffectsMonitor Class Reference

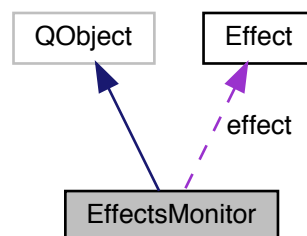
Class for managing effects parameters.

```
#include <EffectsMonitor.h>
```

Inheritance diagram for EffectsMonitor:



Collaboration diagram for EffectsMonitor:



### Public Slots

#### Parameters slots

*User interface functions for effect parameters control.*

- void **updateParameter** (int value)  
*Slot for updating parameters parameters of type int when one of them is changed.*
- void **updateParameter** (double value)  
*Slot for updating parameters parameters of type double when one of them is changed.*
- void **updateParameter** (QString value)  
*Slot for updating parameters of type string when one of them is changed.*
- void **updateParameter** (bool value)  
*Slot for updating parameters parameters of type bool and enum when one of them is changed.*

## Public Member Functions

- [EffectsMonitor](#) (QWidget \*framework)  
*EffectsMonitor constructor.*
- [EffectsMonitor](#) (QWidget \*framework, [Effect](#) \*effect)  
*EffectsMonitor constructor.*
- [~EffectsMonitor](#) ()  
*EffectsMonitor destructor.*
- void [setEffect](#) ([Effect](#) \*effect)  
*It selects an effect.*
- void [clear](#) ()  
*It clears the user interface framework.*
- void [setParameter](#) (std::string key, std::string value)  
*It sets a parameter from the parameter user interface object.*
- void [plotChart](#) ()  
*It plots every chart on the effects monitor.*

## Public Attributes

- [Effect](#) \* [effect](#)
- std::map< [Effect::effectID](#), std::string > [effects](#)
- std::map< [Effect::effectID](#), std::string > [files](#)
- std::map< std::string, std::string > [parameters](#)
- std::map< std::string, [Chart2D](#) \* > [charts](#)

### 6.16.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 33 of file EffectsMonitor.h.

### 6.16.2 Constructor & Destructor Documentation

#### 6.16.2.1 EffectsMonitor() [1/2]

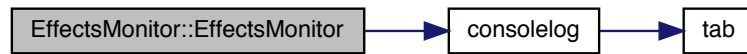
```
EffectsMonitor::EffectsMonitor (
    QWidget * framework )
```

#### Parameters

<i>framework</i>	user interface framework
------------------	--------------------------

Definition at line 10 of file EffectsMonitor.cpp.

Here is the call graph for this function:



### 6.16.2.2 EffectsMonitor() [2/2]

```

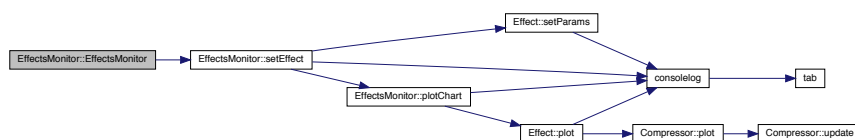
EffectsMonitor::EffectsMonitor (
    QWidget * framework,
    Effect * effect )
  
```

#### Parameters

<i>framework</i>	user interface framework
<i>effect</i>	selected effect to be load

Definition at line 23 of file EffectsMonitor.cpp.

Here is the call graph for this function:



## 6.16.3 Member Function Documentation

### 6.16.3.1 setEffect()

```

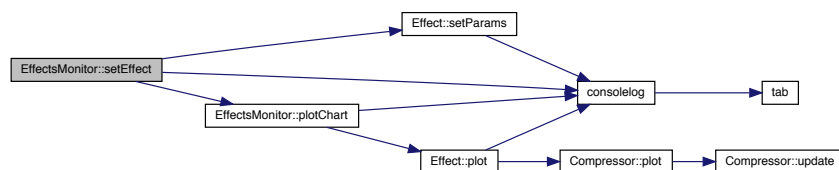
void EffectsMonitor::setEffect (
    Effect * effect )
  
```

#### Parameters

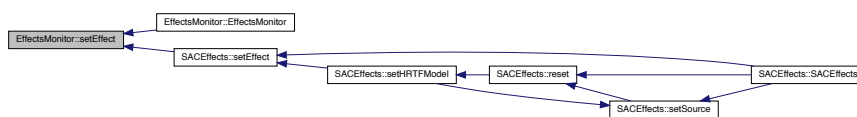
<i>effect</i>	selected effect
---------------	-----------------

Definition at line 40 of file EffectsMonitor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.16.3.2 setParameter()

```

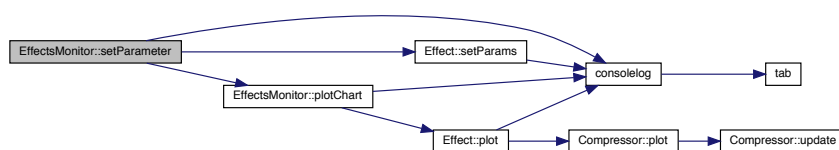
void EffectsMonitor::setParameter (
    std::string parameter,
    std::string value )
  
```

#### Parameters

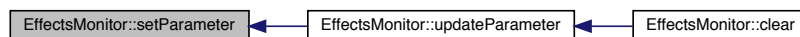
<i>parameter</i>	parameter name
<i>value</i>	new parameter value

Definition at line 362 of file EffectsMonitor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.16.3.3 updateParameter [1/4]

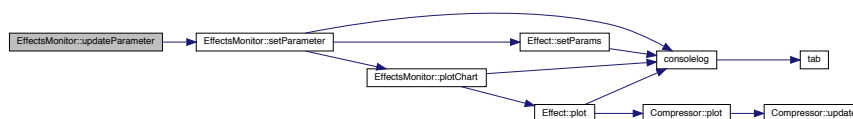
```
void EffectsMonitor::updateParameter (
    int value ) [slot]
```

#### Parameters

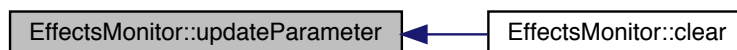
<i>value</i>	changed value
--------------	---------------

Definition at line 403 of file EffectsMonitor.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.16.3.4 updateParameter [2/4]

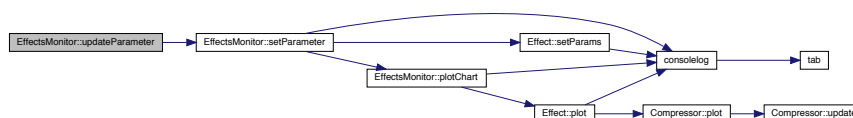
```
void EffectsMonitor::updateParameter (
    double value ) [slot]
```

## Parameters

<i>value</i>	changed value
--------------	---------------

Definition at line 413 of file EffectsMonitor.cpp.

Here is the call graph for this function:



## 6.16.3.5 updateParameter [3/4]

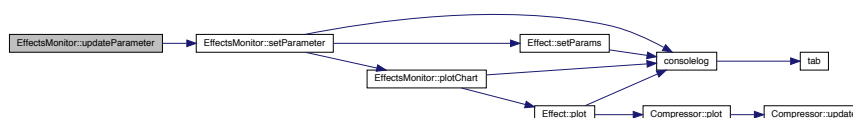
```
void EffectsMonitor::updateParameter (
    QString value ) [slot]
```

## Parameters

<i>value</i>	changed value
--------------	---------------

Definition at line 423 of file EffectsMonitor.cpp.

Here is the call graph for this function:



## 6.16.3.6 updateParameter [4/4]

```
void EffectsMonitor::updateParameter (
    bool value ) [slot]
```

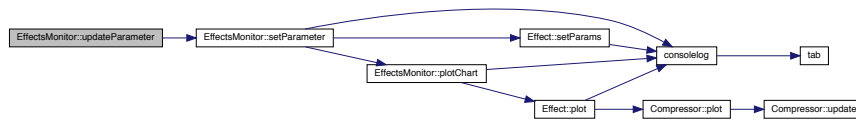
## Parameters

<i>value</i>	changed value
--------------	---------------



Definition at line 433 of file EffectsMonitor.cpp.

Here is the call graph for this function:



## 6.16.4 Member Data Documentation

### 6.16.4.1 charts

```
std::map<std::string, Chart2D *> EffectsMonitor::charts
```

list of charts of effect monitoring

Definition at line 40 of file EffectsMonitor.h.

### 6.16.4.2 effect

```
Effect* EffectsMonitor::effect
```

pointer to current selected effect

Definition at line 36 of file EffectsMonitor.h.

### 6.16.4.3 effects

```
std::map<Effect::effectID, std::string> EffectsMonitor::effects
```

list of all available effects

Definition at line 37 of file EffectsMonitor.h.

#### 6.16.4.4 files

```
std::map<Effect::effectID, std::string> EffectsMonitor::files
```

list of all available effects template files

Definition at line 38 of file EffectsMonitor.h.

#### 6.16.4.5 parameters

```
std::map<std::string, std::string> EffectsMonitor::parameters
```

list of the current effect parameters and their values

Definition at line 39 of file EffectsMonitor.h.

The documentation for this class was generated from the following files:

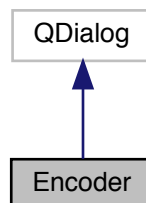
- src/interface/EffectsMonitor.h
- src/interface/EffectsMonitor.cpp

## 6.17 Encoder Class Reference

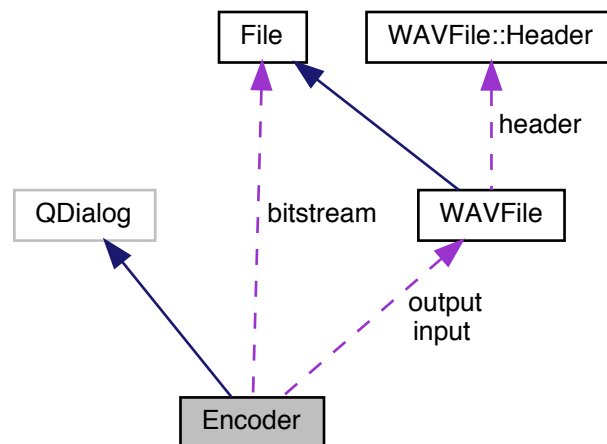
[Encoder](#) window interface.

```
#include <Encoder.h>
```

Inheritance diagram for Encoder:



Collaboration diagram for Encoder:



## Public Member Functions

- [Encoder](#) (QWidget \*parent=0)  
*Encoder constructor.*
- [~Encoder](#) ()  
*Encoder destructor.*
- void [setInput](#) (std::string filename)  
*It sets the input audio file.*
- void [setOutput](#) (std::string filename)  
*It sets the output audio file.*
- void [setTree](#) (int tree)  
*It sets a tree configuration.*

## Public Attributes

- int [fs](#)
- [WAVFile](#) \* [input](#)
- [WAVFile](#) \* [output](#)
- [File](#) \* [bitstream](#)

### 6.17.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 29 of file Encoder.h.

## 6.17.2 Member Function Documentation

### 6.17.2.1 setInput()

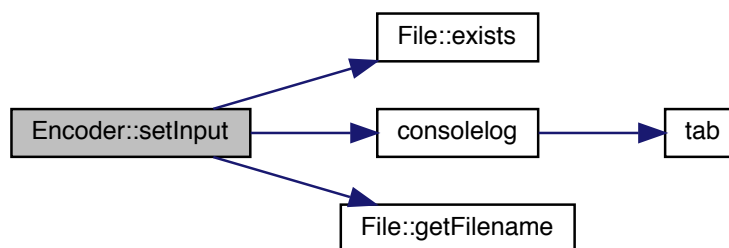
```
void Encoder::setInput (
    std::string filename )
```

#### Parameters

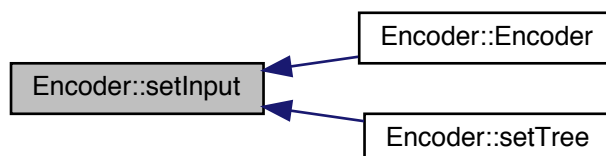
<i>filename</i>	file path
-----------------	-----------

Definition at line 52 of file Encoder.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.17.2.2 setOutput()

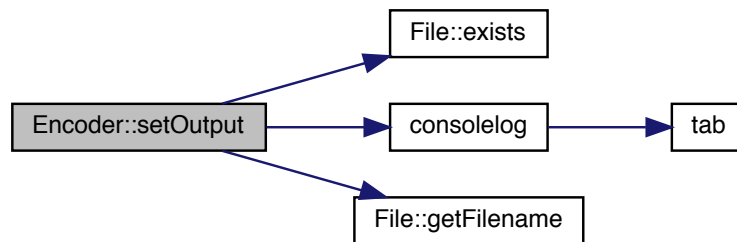
```
void Encoder::setOutput (
    std::string filename )
```

## Parameters

<i>filename</i>	file path
-----------------	-----------

Definition at line 85 of file Encoder.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.17.2.3 setTree()

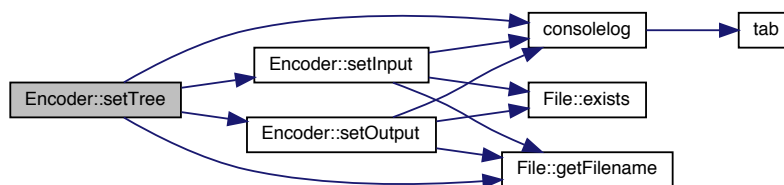
```
void Encoder::setTree (  
    int tree )
```

## Parameters

<i>tree</i>	configuration index
-------------	---------------------

Definition at line 116 of file Encoder.cpp.

Here is the call graph for this function:



### 6.17.3 Member Data Documentation

#### 6.17.3.1 bitstream

`File*` `Encoder::bitstream`

output bit stream file object

Definition at line 35 of file `Encoder.h`.

#### 6.17.3.2 fs

`int` `Encoder::fs`

signal sampling frequency [Hz]

Definition at line 32 of file `Encoder.h`.

#### 6.17.3.3 input

`WAVFile*` `Encoder::input`

input file object

Definition at line 33 of file `Encoder.h`.

#### 6.17.3.4 output

`WAVFile* Encoder::output`

output file object

Definition at line 34 of file Encoder.h.

The documentation for this class was generated from the following files:

- src/interface/Encoder.h
- src/interface/Encoder.cpp

## 6.18 File::Endianness Struct Reference

### Public Types

- enum `endianness` { `littleendian`, `bigendian` }

#### 6.18.1 Detailed Description

Definition at line 23 of file File.h.

#### 6.18.2 Member Enumeration Documentation

##### 6.18.2.1 endianness

enum `File::Endianness::endianness`

##### Enumerator

<code>littleendian</code>	little endian
<code>bigendian</code>	big endian

Definition at line 24 of file File.h.

The documentation for this struct was generated from the following file:

- src/process/File.h

## 6.19 Equalizer Class Reference

Audio equalizer effect.

```
#include <Equalizer.h>
```

## Public Member Functions

- [Equalizer](#) ()  
*[Equalizer](#) constructor.*
- void [apply](#) (float \*\*input, float \*\*output, int samples, std::vector< [SACBitstream::ChannelType::channeltype](#) > channels)  
*It applies equalization effect.*
- void [peakingFilter](#) (float \*input, float \*output, int samples, double f\_0, double gain, double Q, int order)  
*It applies a peaking filter.*
- void [lowShelfFilter](#) (float \*input, float \*output, int samples, double f\_0, double gain, int order)  
*It applies a low shelf filter.*
- void [highShelfFilter](#) (float \*input, float \*output, int samples, double f\_0, double gain, int order)  
*It applies a high shelf filter.*
- void [filter](#) (float \*x, float \*y, int samples, float \*a, float \*b, int order)  
*It applies a filter according to the transfer function  $H(z) = (b[0] + b[1] \cdot z^{-1} + \dots + b[\text{order}] \cdot z^{-\text{order}}) / (a[0] + a[1] \cdot z^{-1} + \dots + a[\text{order}] \cdot z^{-\text{order}})$ .*

### 6.19.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 12 of file Equalizer.h.

### 6.19.2 Member Function Documentation

#### 6.19.2.1 apply()

```
void Equalizer::apply (
    float ** input,
    float ** output,
    int samples,
    std::vector< SACBitstream::ChannelType::channeltype > channels )
```

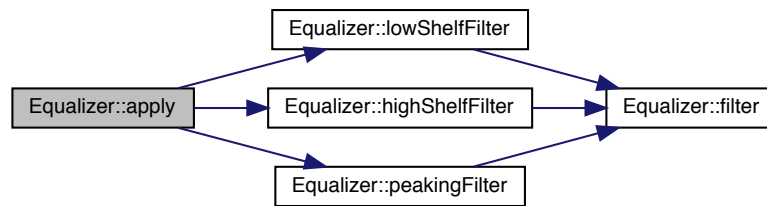
#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>channels</i>	vector of channel types

Definition at line 16 of file Equalizer.cpp.



Here is the call graph for this function:



Here is the caller graph for this function:



### 6.19.2.2 filter()

```

void Equalizer::filter (
    float * input,
    float * output,
    int samples,
    float * a,
    float * b,
    int order )
  
```

#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>a</i>	y coefficients of transfer function
<i>b</i>	x coefficients of transfer function
<i>order</i>	filter order (value of the highest exponent)

Definition at line 157 of file Equalizer.cpp.

Here is the caller graph for this function:



### 6.19.2.3 highShelfFilter()

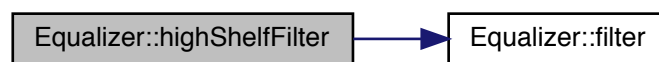
```
void Equalizer::highShelfFilter (
    float * input,
    float * output,
    int samples,
    double f_0,
    double gain,
    int order )
```

#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>f_0</i>	midpoint frequency (real frequency / sampling frequency)
<i>gain</i>	peak power gain
<i>order</i>	filter order (value of the highest exponent)

Definition at line 126 of file Equalizer.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.19.2.4 lowShelfFilter()

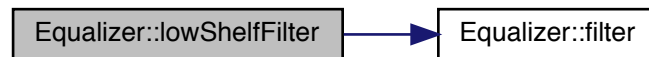
```
void Equalizer::lowShelfFilter (
    float * input,
    float * output,
    int samples,
    double f_0,
    double gain,
    int order )
```

## Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>f_0</i>	midpoint frequency (real frequency / sampling frequency)
<i>gain</i>	peak power gain
<i>order</i>	filter order (value of the highest exponent)

Definition at line 95 of file Equalizer.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.19.2.5 peakingFilter()

```

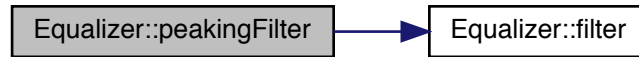
void Equalizer::peakingFilter (
    float * input,
    float * output,
    int samples,
    double f_0,
    double gain,
    double Q,
    int order )
  
```

## Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>f_0</i>	center frequency (real frequency / sampling frequency)
<i>gain</i>	peak power gain
<i>Q</i>	quality factor
<i>order</i>	filter order (value of the highest exponent)

Definition at line 64 of file Equalizer.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

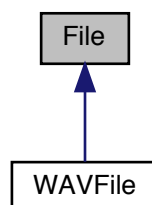
- `src/effects/Equalizer.h`
- `src/effects/Equalizer.cpp`

## 6.20 File Class Reference

Audio file class.

```
#include <File.h>
```

Inheritance diagram for File:



### Classes

- struct [Endianess](#)

## Endianness

[Endianness](#) type.

- [File](#) (bool writepermission)  
*File constructor.*
- [File](#) (std::string filename, bool writepermission)  
*File constructor.*
- [~File](#) ()  
*File destructor.*
- void [setFilename](#) (std::string filename)  
*It sets the file path name.*
- std::string [getFilename](#) ()  
*It gets the file path name.*
- void [setCursor](#) (int cursor)  
*It sets the file reading cursor to keep on reading from another position.*
- int [getCursor](#) ()  
*It gets the current file reading cursor.*
- int [size](#) ()  
*It gets the total file size.*
- bool [exists](#) ()  
*It indicates if the file object exists.*
- char \* [read](#) (int length)  
*It reads data from the file.*
- void [write](#) (const char \*data, int length)  
*It writes data on the file.*
- std::string [readText](#) (int length)  
*It reads text data from the file.*
- void [writeText](#) (std::string data)  
*It writes text data on the file.*
- unsigned [readNumber](#) (int length, [Endianness::endianness](#) endianness)  
*It reads a data number from the file.*
- void [writeNumber](#) (unsigned int data, int length, [Endianness::endianness](#) endianness)  
*It writes a data number on the file.*

### 6.20.1 Detailed Description

Author

Andrés González Fornell

Definition at line 17 of file File.h.

### 6.20.2 Constructor & Destructor Documentation

#### 6.20.2.1 File()

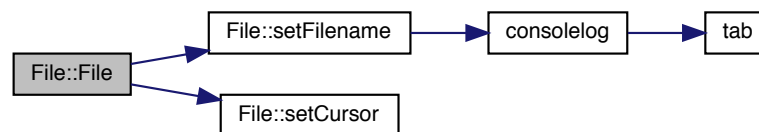
```
File::File (
    std::string filename,
    bool writepermission )
```

## Parameters

<i>filename</i>	file path
<i>writepermission</i>	file write permission (true if it is allowed)

Definition at line 17 of file File.cpp.

Here is the call graph for this function:



## 6.20.3 Member Function Documentation

### 6.20.3.1 exists()

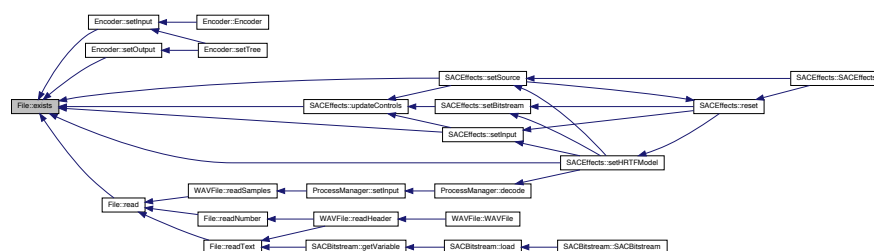
```
bool File::exists ( )
```

## Returns

true if the file object exists

Definition at line 92 of file File.cpp.

Here is the caller graph for this function:



6.20.3.2 `getCursor()`

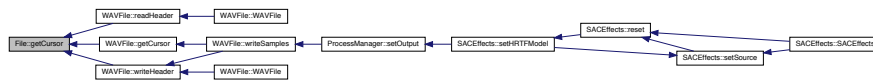
```
int File::getCursor ( )
```

**Returns**

cursor [Bytes] from the beginning of the file

Definition at line 70 of file File.cpp.

Here is the caller graph for this function:

6.20.3.3 `getFilename()`

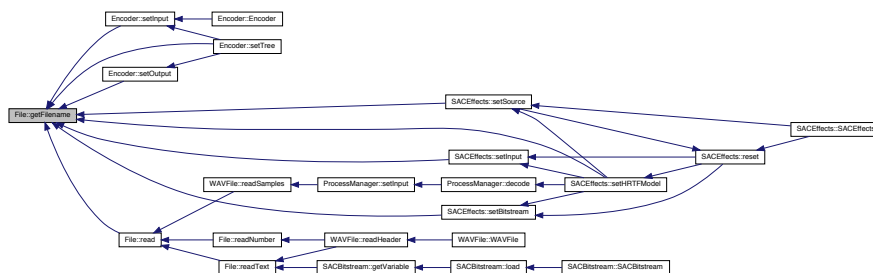
```
std::string File::getFilename ( )
```

**Returns**

file path name

Definition at line 53 of file File.cpp.

Here is the caller graph for this function:

6.20.3.4 `read()`

```
char * File::read (
    int length )
```

## Parameters

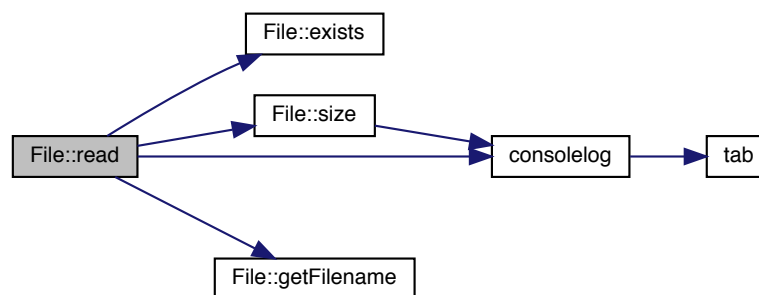
<i>length</i>	data length [Bytes]
---------------	---------------------

## Returns

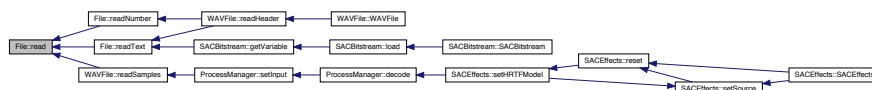
data pointer

Definition at line 105 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.20.3.5 readNumber()

```

unsigned int File::readNumber (
    int length,
    Endianness::endianness endianness )

```

## Parameters

<i>length</i>	data length [Bytes]
<i>endianness</i>	data order (big endian or little endian)

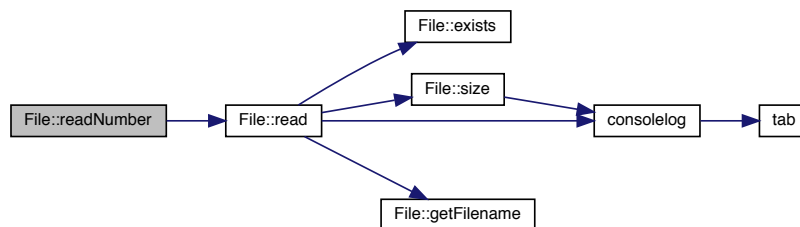


**Returns**

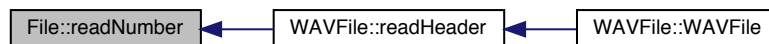
value of data number

Definition at line 175 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:

**6.20.3.6 readText()**

```
std::string File::readText (
    int length )
```

**Parameters**

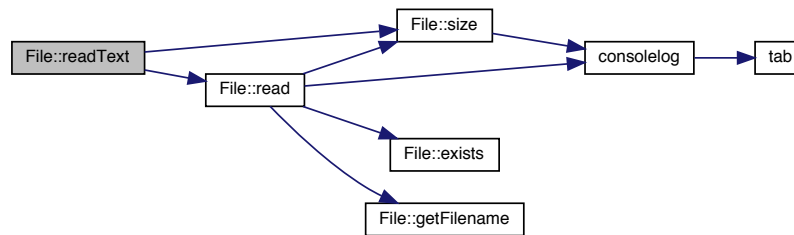
<i>length</i>	data length [Bytes] (if length = 0 function returns all available data from the file)
---------------	---

**Returns**

string of data

Definition at line 145 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.20.3.7 setCursor()

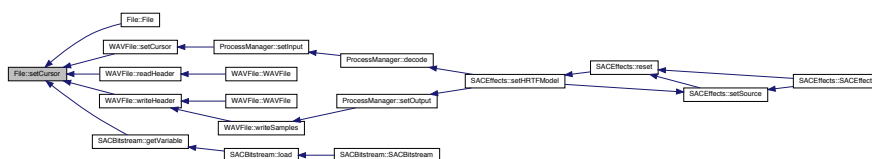
```
void File::setCursor (
    int cursor )
```

#### Parameters

<i>cursor</i>	new cursor position [Bytes] from the beginning of the file
---------------	--

Definition at line 61 of file File.cpp.

Here is the caller graph for this function:



### 6.20.3.8 setFilename()

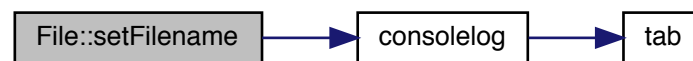
```
void File::setFilename (
    std::string filename )
```

#### Parameters

<i>filename</i>	file path name
-----------------	----------------

Definition at line 35 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.20.3.9 size()

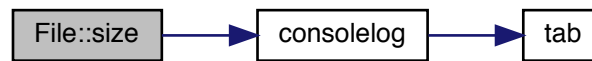
```
int File::size ( )
```

#### Returns

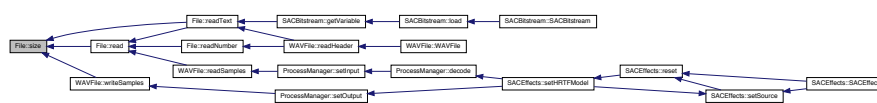
file size [Bytes]

Definition at line 78 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.20.3.10 write()

```

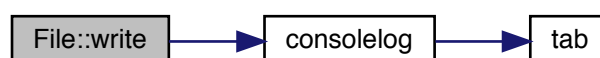
void File::write (
    const char * data,
    int length )
  
```

##### Parameters

<i>data</i>	data pointer
<i>length</i>	data length [Bytes]

Definition at line 131 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.20.3.11 writeNumber()

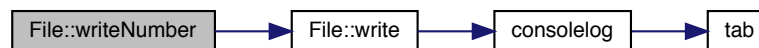
```
void File::writeNumber (
    unsigned int value,
    int length,
    Endianness::endianness endianness )
```

#### Parameters

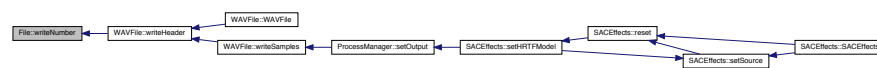
<i>value</i>	value of data number
<i>length</i>	data length [Bytes]
<i>endianness</i>	data order (big endian or little endian)

Definition at line 199 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.20.3.12 writeText()

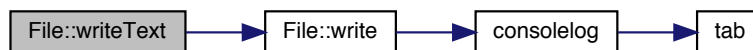
```
void File::writeText (
    std::string data )
```

## Parameters

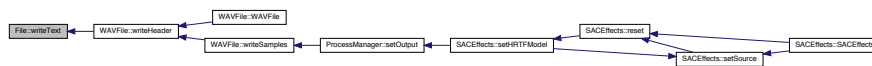
<i>data</i>	string of data
-------------	----------------

Definition at line 165 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- src/process/File.h
- src/process/File.cpp

## 6.21 WAVFile::Header Struct Reference

Audio file header struct.

```
#include <File.h>
```

### Public Member Functions

- int **size** ()

## Public Attributes

### Chunk header

*It indicates the audio format (wave).*

- std::string [chunkID](#)
- unsigned int [chunksizesize](#)
- std::string [format](#)

### Subshunk 1 header

*It describes the format of the sound information in the data sub-chunk.*

- std::string [subchunk1ID](#)
- unsigned int [subchunk1size](#)
- unsigned int [audioformat](#)
- unsigned int [numchannels](#)
- unsigned int [samplerate](#)
- unsigned int [byterate](#)
- unsigned int [blockalign](#)
- unsigned int [bitspersample](#)

### Subshunk 2 header

*It indicates the size of the sound information.*

- std::string [subchunk2ID](#)
- unsigned int [subchunk2size](#)

## 6.21.1 Detailed Description

Definition at line 62 of file File.h.

## 6.21.2 Member Data Documentation

### 6.21.2.1 audioformat

```
unsigned int WAVFile::Header::audioformat
```

PCM = 1 (linear quantization) values others than 1 indicate some form of compression

Definition at line 79 of file File.h.

### 6.21.2.2 bitspersample

```
unsigned int WAVFile::Header::bitspersample
```

number of bits per sample

Definition at line 84 of file File.h.

#### 6.21.2.3 blockalign

```
unsigned int WAVFile::Header::blockalign
```

number of bytes for one sample including all channels (= numchannels \* bitspersample/8)

Definition at line 83 of file File.h.

#### 6.21.2.4 byterate

```
unsigned int WAVFile::Header::byterate
```

byte rate (= samplerate \* numchannels \* bitspersample/8)

Definition at line 82 of file File.h.

#### 6.21.2.5 chunkID

```
std::string WAVFile::Header::chunkID
```

it contains the letters "RIFF" in ASCII form

Definition at line 68 of file File.h.

#### 6.21.2.6 chunksize

```
unsigned int WAVFile::Header::chunksize
```

size of the entire file in bytes minus 8 bytes for the two fields not included in this count (ChunkID and ChunkSize)

Definition at line 69 of file File.h.

#### 6.21.2.7 format

```
std::string WAVFile::Header::format
```

it contains the letters "WAVE"

Definition at line 70 of file File.h.



#### 6.21.2.8 numchannels

```
unsigned int WAVFile::Header::numchannels
```

number of channels

Definition at line 80 of file File.h.

#### 6.21.2.9 samplerate

```
unsigned int WAVFile::Header::samplerate
```

sample rate

Definition at line 81 of file File.h.

#### 6.21.2.10 subchunk1ID

```
std::string WAVFile::Header::subchunk1ID
```

it contains the letters "fmt "

Definition at line 77 of file File.h.

#### 6.21.2.11 subchunk1size

```
unsigned int WAVFile::Header::subchunk1size
```

size of the rest of the subchunk (16 for PCM)

Definition at line 78 of file File.h.

#### 6.21.2.12 subchunk2ID

```
std::string WAVFile::Header::subchunk2ID
```

it contains the letters "data"

Definition at line 91 of file File.h.

### 6.21.2.13 subchunk2size

```
unsigned int WAVFile::Header::subchunk2size
```

size of ther rest of the subchunk (it is the size of the data)

Definition at line 92 of file File.h.

The documentation for this struct was generated from the following file:

- src/process/File.h

## 6.22 HRTFModel Struct Reference

SAC decoder parameter HRTF model.

```
#include <SACEffects.h>
```

### Public Types

- enum [hrtfmodel](#) { [kemar](#) = 0, [vast](#) = 1, [mps\\_vt](#) = 2 }

### 6.22.1 Detailed Description

Definition at line 55 of file SACEffects.h.

### 6.22.2 Member Enumeration Documentation

#### 6.22.2.1 hrtfmodel

```
enum HRTFModel::hrtfmodel
```

#### Enumerator

kemar	kemar head related transfer funcion model
vast	vast head related transfer function model
mps_vt	MPS VT head related transfer funcion model

Definition at line 56 of file SACEffects.h.

The documentation for this struct was generated from the following file:

- src/interface/SACEffects.h

## 6.23 LogType Struct Reference

### Public Types

- enum `logtype` {  
    `info`, `warning`, `error`, `progress`,  
    `interaction` }

#### 6.23.1 Detailed Description

Definition at line 13 of file `Logger.h`.

#### 6.23.2 Member Enumeration Documentation

##### 6.23.2.1 `logtype`

```
enum LogType::logtype
```

##### Enumerator

<code>info</code>	The message is not important, just some information for the user
<code>warning</code>	The message is a warning
<code>error</code>	The message comes from an bad execution (do not confuse with execution or compilation errors)
<code>progress</code>	Information about the current steps in the running execution
<code>interaction</code>	Information about an user interaction

Definition at line 14 of file `Logger.h`.

The documentation for this struct was generated from the following file:

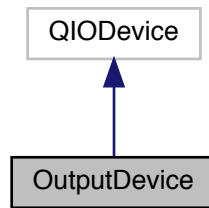
- `src/tools/Logger.h`

## 6.24 OutputDevice Class Reference

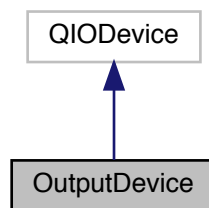
Audio output device class (QIODevice extension).

```
#include <AudioOutput.h>
```

Inheritance diagram for OutputDevice:



Collaboration diagram for OutputDevice:



## Public Member Functions

- [OutputDevice](#) (QAudioFormat format)  
*OutputDevice* constructor.
- [~OutputDevice](#) ()  
*OutputDevice* destructor.
- void [send](#) (float \*signal, int samples)  
*It sends an audio signal to the buffer to be sent to the audio output device.*
- qint64 [readData](#) (char \*data, qint64 length)  
*It gets data from the audio output device.*
- qint64 [writeData](#) (const char \*data, qint64 length)  
*It gets written data from the audio input device (not used).*
- qint64 [bytesAvailable](#) () const  
*It gets available bytes to be read by the audio output device.*
- void [test](#) (double amplitude, double frequency, float duration)  
*It plays an audio test by generating a tone.*
- void [clear](#) ()  
*It clears output buffer.*

## Public Attributes

- char \* [buffer](#)
- int [cursor\\_read](#)
- int [cursor\\_write](#)
- int [buffersize](#)

### 6.24.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 32 of file `AudioOutput.h`.

### 6.24.2 Constructor & Destructor Documentation

#### 6.24.2.1 `OutputDevice()`

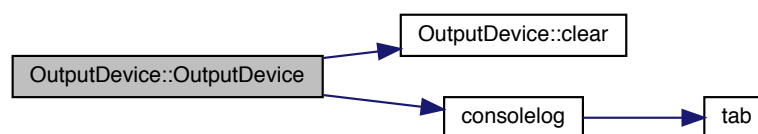
```
OutputDevice::OutputDevice (
    QAudioFormat format )
```

#### Parameters

<i>format</i>	audio format object
---------------	---------------------

Definition at line 155 of file `AudioOutput.cpp`.

Here is the call graph for this function:



### 6.24.3 Member Function Documentation

#### 6.24.3.1 bytesAvailable()

```
qint64 OutputDevice::bytesAvailable ( ) const
```

##### Returns

Definition at line 265 of file AudioOutput.cpp.

#### 6.24.3.2 readData()

```
qint64 OutputDevice::readData (
    char * data,
    qint64 length )
```

##### Parameters

<i>data</i>	data pointer
<i>length</i>	data length

##### Returns

Definition at line 228 of file AudioOutput.cpp.

#### 6.24.3.3 send()

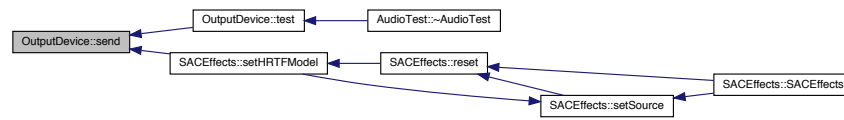
```
void OutputDevice::send (
    float * signal,
    int samples )
```

##### Parameters

<i>signal</i>	audio signal pointer
<i>samples</i>	number of samples

Definition at line 195 of file AudioOutput.cpp.

Here is the caller graph for this function:



#### 6.24.3.4 test()

```
void OutputDevice::test (
    double amplitude,
    double frequency,
    float duration )
```

##### Parameters

<i>amplitude</i>	tone amplitude (from 0 to 1)
<i>frequency</i>	tone frequency [Hz]
<i>duration</i>	test duration [s]

Definition at line 284 of file AudioOutput.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.24.3.5 writeData()

```
qint64 OutputDevice::writeData (
    const char * data,
    qint64 length )
```

##### Parameters

<i>data</i>	data pointer
<i>length</i>	data length

##### Returns

Definition at line 255 of file AudioOutput.cpp.

### 6.24.4 Member Data Documentation

#### 6.24.4.1 buffer

```
char* OutputDevice::buffer
```

audio output data buffer

Definition at line 35 of file AudioOutput.h.

#### 6.24.4.2 buffersize

```
int OutputDevice::buffersize
```

total size of buffer [Bytes]

Definition at line 38 of file AudioOutput.h.

#### 6.24.4.3 cursor\_read

```
int OutputDevice::cursor_read
```

cursor of read audio output data in buffer

Definition at line 36 of file AudioOutput.h.



## 6.24.4.4 cursor\_write

```
int OutputDevice::cursor_write
```

cursor of pending audio output data in buffer

Definition at line 37 of file AudioOutput.h.

The documentation for this class was generated from the following files:

- src/interface/AudioOutput.h
- src/interface/AudioOutput.cpp

## 6.25 Panning Class Reference

Audio panning effect.

```
#include <Panning.h>
```

### Public Member Functions

- [Panning](#) ()  
*Panning constructor.*
- void [apply](#) (float \*\*input, float \*\*output, int samples, std::vector< [SACBitstream::ChannelType::channeltype](#) > channels)  
*It applies panning effect.*

### 6.25.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 12 of file Panning.h.

### 6.25.2 Member Function Documentation

#### 6.25.2.1 apply()

```
void Panning::apply (
    float ** input,
    float ** output,
    int samples,
    std::vector< SACBitstream::ChannelType::channeltype > channels )
```

## Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>channels</i>	vector of channel types

Definition at line 16 of file Panning.cpp.

Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- src/effects/Panning.h
- src/effects/Panning.cpp

## 6.26 ProcessManager Class Reference

Process manager class. It contains all functions to perform the signal treatment process.

```
#include <ProcessManager.h>
```

### Public Member Functions

- [ProcessManager](#) (int chunksize)  
*ProcessManager constructor.*
- [~ProcessManager](#) ()  
*ProcessManager destructor.*
- bool [setInput](#) (std::string filename)  
*It sets input variable from the existing input file.*
- bool [setOutput](#) (std::string filename)  
*It sets an output file from the existing output variable.*
- bool [decode](#) (std::string [input](#), std::string bitstream, std::string [output](#), int upmixtype, int decodingtype, int binauralquality, int hrtfmodel)  
*It performs the SAC encoder.*
- bool [applyEffect](#) ([Effect](#) \*effect, std::vector< bool > [channels](#), std::vector< double > levels)  
*It applies the selected effect to the input stream.*
- void [clear](#) ()  
*It clears all variables and resets the process.*

## Public Attributes

- int [fs](#)
- float \*\* [input](#)
- float \*\* [output](#)
- int [channels](#)
- int [samples](#)
- int [cursor](#)
- int [total](#)
- float \*\* [NRG](#)
- float \*\* [OLD](#)

### 6.26.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 21 of file ProcessManager.h.

### 6.26.2 Constructor & Destructor Documentation

#### 6.26.2.1 ProcessManager()

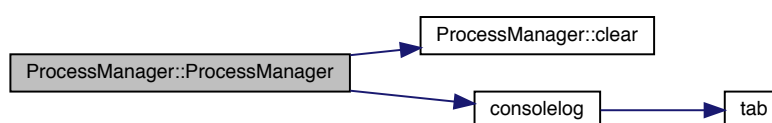
```
ProcessManager::ProcessManager (
    int chunksize )
```

#### Parameters

<i>chunksize</i>	number of samples in a chunk to apply effect step by step (if 0 then chunk size is the number of samples and effect is applied at once)
------------------	---

Definition at line 8 of file ProcessManager.cpp.

Here is the call graph for this function:



## 6.26.3 Member Function Documentation

### 6.26.3.1 applyEffect()

```
bool ProcessManager::applyEffect (
    Effect * effect,
    std::vector< bool > channels,
    std::vector< double > levels )
```

#### Parameters

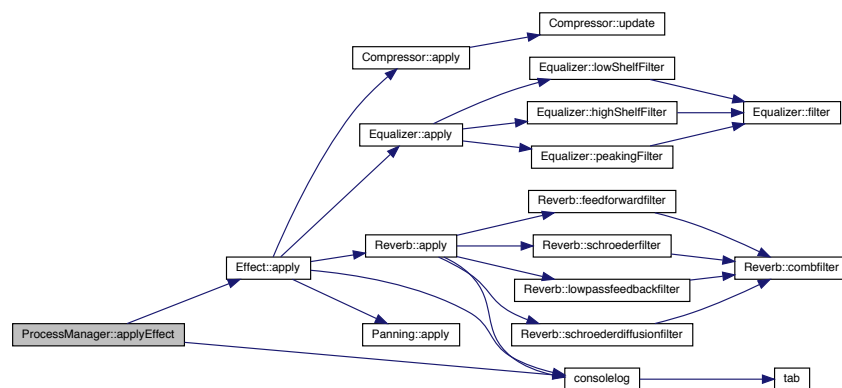
<i>effect</i>	effect object (it includes all parameters)
<i>channels</i>	boolean vector where true means to apply effect to that channel
<i>levels</i>	vector of input levels ( $\geq 0$ ) for each channel

#### Returns

true if it was successful

Definition at line 135 of file ProcessManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.26.3.2 decode()

```
bool ProcessManager::decode (
    std::string input,
    std::string bitstream,
    std::string output,
    int decodingtype,
    int upmixtype,
    int binauralquality,
    int hrtfmodel )
```

## Parameters

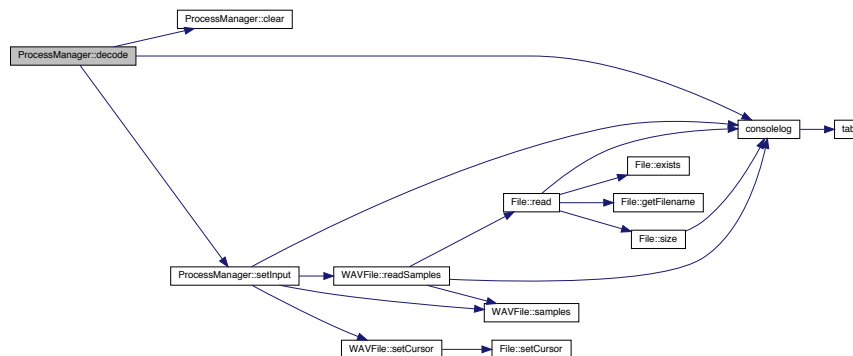
<i>input</i>	filename of the multichannel input audio file
<i>output</i>	filename of the downmix output audio file (it will be automatically created)
<i>bitstream</i>	filename of the bitstream output file or "buried" (it will be automatically created)
<i>upmixtype</i>	upmix type 0: normal 1: blind 2: binaural 3: stereo
<i>decodingtype</i>	decoding type 0: low 1: high
<i>binauralquality</i>	binaural upmix quality 0: parametric 1: filtering
<i>hrtfmodel</i>	HRTF model 0: kemar 1: vast 2: mps_vt

## Returns

true if it was successful

Definition at line 80 of file ProcessManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.26.3.3 setInput()

```
bool ProcessManager::setInput (
    std::string filename )
```

#### Parameters

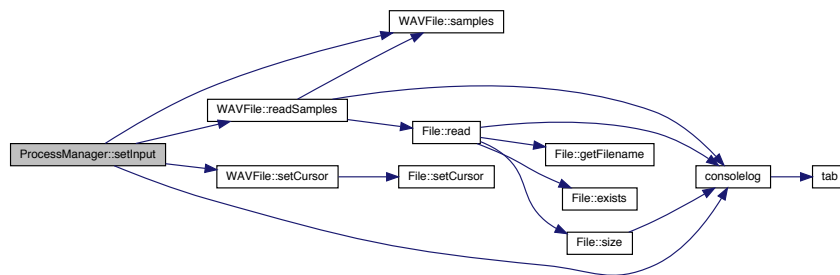
<i>filename</i>	audio input file name
-----------------	-----------------------

#### Returns

true if it was successful

Definition at line 30 of file ProcessManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.26.3.4 setOutput()

```
bool ProcessManager::setOutput (
    std::string filename )
```

#### Parameters

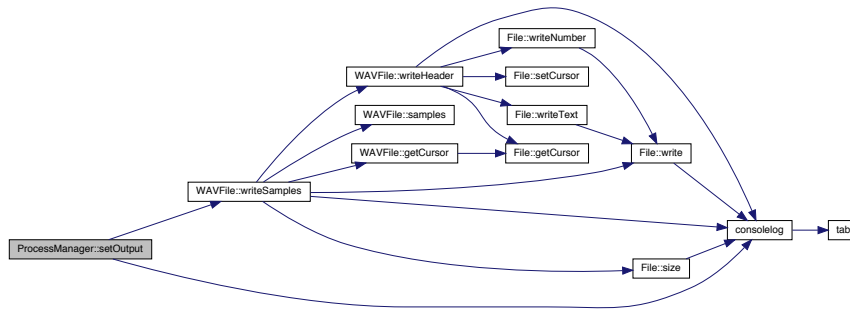
<i>filename</i>	audio output file name
-----------------	------------------------

**Returns**

true if it was successful

Definition at line 60 of file ProcessManager.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.26.4 Member Data Documentation

### 6.26.4.1 channels

```
int ProcessManager::channels
```

number of channels

Definition at line 26 of file ProcessManager.h.

### 6.26.4.2 cursor

```
int ProcessManager::cursor
```

pointer to current sample index when executing real time process

Definition at line 28 of file ProcessManager.h.

#### 6.26.4.3 fs

```
int ProcessManager::fs
```

signal sampling frequency

Definition at line 23 of file ProcessManager.h.

#### 6.26.4.4 input

```
float** ProcessManager::input
```

vector of input channels stream (sample = input[channel][sample index])

Definition at line 24 of file ProcessManager.h.

#### 6.26.4.5 NRG

```
float** ProcessManager::NRG
```

object energies (NRG[channel][chunk])

Definition at line 31 of file ProcessManager.h.

#### 6.26.4.6 OLD

```
float** ProcessManager::OLD
```

object level differences (OLD[channel][chunk])

Definition at line 32 of file ProcessManager.h.

#### 6.26.4.7 output

```
float** ProcessManager::output
```

vector of input channels stream (sample = output[channel][sample index])

Definition at line 25 of file ProcessManager.h.



## 6.26.4.8 samples

```
int ProcessManager::samples
```

number of samples in each channel

Definition at line 27 of file ProcessManager.h.

## 6.26.4.9 total

```
int ProcessManager::total
```

number of available output samples

Definition at line 29 of file ProcessManager.h.

The documentation for this class was generated from the following files:

- src/process/ProcessManager.h
- src/process/ProcessManager.cpp

## 6.27 Reverb Class Reference

Audio reverb effect.

```
#include <Reverb.h>
```

### Public Member Functions

- [Reverb\(\)](#)  
*Reverb constructor.*
- void [apply](#) (float \*input, float \*output, int samples, std::vector< [SACBitstream::ChannelType::channeltype](#) > channels)  
*It applies reverb effect.*
- void [schroederfilter](#) (float \*input, float \*output, int samples, bool addition, float gain, float g, int delay)  
*It applies a schroeder allpass filter, according to the transfer function  $H(z) = gain * (g + z^{-delay}) / (1 + g \cdot z^{-delay})$ .*
- void [schroederdiffusionfilter](#) (float \*input, float \*output, int samples, bool addition, float gain, float g, int delay)  
*It applies a schroeder diffusion allpass filter, according to the transfer function  $H(z) = gain * (-g + z^{-delay}) / (1 - g \cdot z^{-delay})$ .*
- void [feedforwardfilter](#) (float \*input, float \*output, int samples, bool addition, float gain, float original, int delay)  
*It applies a feed forward comb filter, according to the transfer function  $H(z) = gain * (g + z^{-delay})$ .*
- void [lowpassfeedbackfilter](#) (float \*input, float \*output, int samples, bool addition, float gain, float rs, float d, int delay)  
*It applies a Schroeder-Moorer low pass feedback comb filter, according to the transfer function  $H(z) = gain / (1 - f \cdot (1-d) / (1-d \cdot z^{-1}) \cdot z^{-N})$ .*
- void [combfilter](#) (float \*input, float \*output, int samples, bool addition, float \*a, float \*b, int order, float a\_delay, float b\_delay, int delay)  
*It applies a comb filter, according to the transfer function  $H(z) = (b[0] + b[1] \cdot z^{-1} + \dots + b[order] \cdot z^{-order} + b_{\leftarrow delay} \cdot z^{-delay}) / (a[0] + a[1] \cdot z^{-1} + \dots + a[order] \cdot z^{-order} + a_{\leftarrow delay} \cdot z^{-delay})$ .*

## 6.27.1 Detailed Description

### Author

Andrés González Fornell

Definition at line 12 of file Reverb.h.

## 6.27.2 Member Function Documentation

### 6.27.2.1 apply()

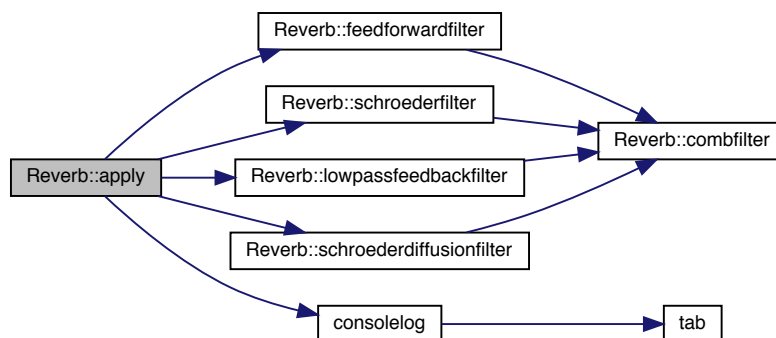
```
void Reverb::apply (
    float ** input,
    float ** output,
    int samples,
    std::vector< SACBitstream::ChannelType::channeltype > channels )
```

#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>channels</i>	vector of channel types

Definition at line 21 of file Reverb.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.27.2.2 combfilter()

```

void Reverb::combfilter (
    float * input,
    float * output,
    int samples,
    bool addition,
    float * a,
    float * b,
    int order,
    float a_delay,
    float b_delay,
    int delay )
  
```

#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>addition</i>	true if the filter is in serie (filter output is summed to the existing output samples) and false if the filter is in cascade (filter output overwrites the existing output samples)
<i>a</i>	y coefficients of transfer function
<i>b</i>	x coefficients of transfer function
<i>order</i>	filter order (value of the highest exponent) without including delay term
<i>a_delay</i>	y delay coefficients of transfer function
<i>b_delay</i>	x delay coefficients of transfer function
<i>delay</i>	number of samples to delay

Definition at line 179 of file Reverb.cpp.

Here is the caller graph for this function:



### 6.27.2.3 feedforwardfilter()

```
void Reverb::feedforwardfilter (
    float * input,
    float * output,
    int samples,
    bool addition,
    float gain,
    float original,
    int delay )
```

#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>addition</i>	true if the filter is in serie (filter output is summed to the existing output samples) and false if the filter is in cascade (filter output overwrites the existing output samples)
<i>gain</i>	function transfer gain
<i>original</i>	gain of the original signal
<i>delay</i>	number of samples to delay

Definition at line 119 of file Reverb.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.27.2.4 lowpassfeedbackfilter()

```
void Reverb::lowpassfeedbackfilter (
    float * input,
    float * output,
    int samples,
```

```

    bool addition,
    float gain,
    float rs,
    float d,
    int delay )

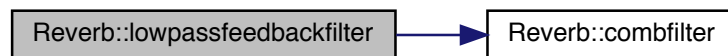
```

#### Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>addition</i>	true if the filter is in serie (filter output is summed to the existing output samples) and false if the filter is in cascade (filter output overwrites the existing output samples)
<i>gain</i>	function transfer gain
<i>rs</i>	feed forward comb filter gain
<i>d</i>	low pass filter gain
<i>delay</i>	number of samples to delay

Definition at line 147 of file Reverb.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.27.2.5 schroederdiffusionfilter()

```

void Reverb::schroederdiffusionfilter (
    float * input,
    float * output,
    int samples,
    bool addition,
    float gain,
    float g,
    int delay )

```

## Parameters

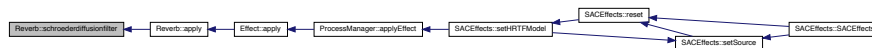
<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>addition</i>	true if the filter is in serie (filter output is summed to the existing output samples) and false if the filter is in cascade (filter output overwrites the existing output samples)
<i>gain</i>	function transfer gain
<i>g</i>	all pass filter gain
<i>delay</i>	number of samples to delay

Definition at line 92 of file Reverb.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.27.2.6 schroederfilter()

```

void Reverb::schroederfilter (
    float * input,
    float * output,
    int samples,
    bool addition,
    float gain,
    float g,
    int delay )
  
```

## Parameters

<i>input</i>	input signal pointer
<i>output</i>	output signal pointer
<i>samples</i>	number of samples
<i>addition</i>	true if the filter is in serie (filter output is summed to the existing output samples) and false if the filter is in cascade (filter output overwrites the existing output samples)
<i>gain</i>	function transfer gain
<i>g</i>	all pass filter gain
<i>delay</i>	number of samples to delay

Definition at line 65 of file Reverb.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



The documentation for this class was generated from the following files:

- `src/effects/Reverb.h`
- `src/effects/Reverb.cpp`

## 6.28 SACBitstream Class Reference

SAC bitstream class.

```
#include <SACBitstream.h>
```

### Classes

- struct [ChannelType](#)  
*It specifies the channel type.*

### Public Member Functions

- [SACBitstream](#) ()  
*Bitstream constructor.*
- [SACBitstream](#) (std::string filename)  
*Bitstream constructor.*
- [~SACBitstream](#) ()  
*Bitstream destructor.*
- long [getVariable](#) (int position, int length)  
*It gets the value of a bitstream variable.*
- void [load](#) ()  
*It loads variables from bitstream file.*

## Public Attributes

- int `fs`
- `std::vector< ChannelType::channeltype >` `channels`
- double `gain_surround`
- double `gain_LFE`
- double `gain_downmix`

## 6.28.1 Detailed Description

### Author

Andrés González Fornell

Definition at line 16 of file SACBitstream.h.

## 6.28.2 Member Function Documentation

### 6.28.2.1 `getVariable()`

```
long int SACBitstream::getVariable (
    int position,
    int length )
```

#### Parameters

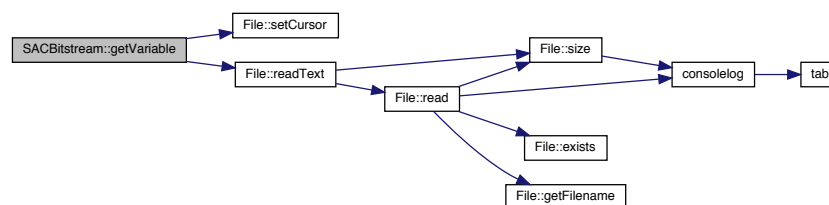
<i>position</i>	position in bits
<i>length</i>	number of bits

#### Returns

value of the bitstream variable

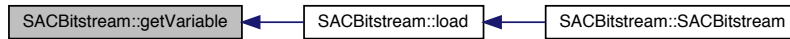
Definition at line 43 of file SACBitstream.cpp.

Here is the call graph for this function:





Here is the caller graph for this function:



### 6.28.3 Member Data Documentation

#### 6.28.3.1 channels

```
std::vector<ChannelType::channeltype> SACBitstream::channels
```

channels order

Definition at line 36 of file `SACBitstream.h`.

#### 6.28.3.2 fs

```
int SACBitstream::fs
```

signal sampling frequencye

Definition at line 35 of file `SACBitstream.h`.

#### 6.28.3.3 gain\_downmix

```
double SACBitstream::gain_downmix
```

gain of downmix

Definition at line 39 of file `SACBitstream.h`.

#### 6.28.3.4 gain\_LFE

```
double SACBitstream::gain_LFE
```

downmix of LFE channels

Definition at line 38 of file `SACBitstream.h`.

### 6.28.3.5 gain\_surround

```
double SACBitstream::gain_surround
```

downmix of surround channels

Definition at line 37 of file SACBitstream.h.

The documentation for this class was generated from the following files:

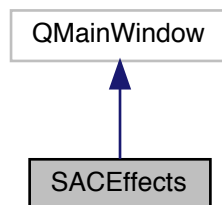
- src/sac/SACBitstream.h
- src/sac/SACBitstream.cpp

## 6.29 SACEffects Class Reference

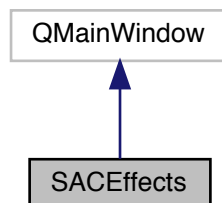
[SACEffects](#) window interface.

```
#include <SACEffects.h>
```

Inheritance diagram for SACEffects:



Collaboration diagram for SACEffects:



## Public Member Functions

- [SACEffects](#) (QWidget \*framework=0)  
*SACEffects constructor.*
- [~SACEffects](#) ()  
*SACEffects destructor.*
- void [play](#) ()  
*It starts playing input.*
- void [pause](#) ()  
*It pauses input playback.*
- void [reset](#) ()  
*It resets all decoding parameters, including input file.*
- void [updateControls](#) ()  
*It updates enableity of user interface controls according to the current parameters state.*
- void [setEffect](#) ([Effect::effectID](#) effect)  
*It sets an effect for the effect monitor.*
- void [setSource](#) (std::string filename)  
*It sets the source audio file.*
- void [setBitstream](#) (std::string filename)  
*It sets the bitstream audio file.*
- void [setInput](#) (std::string filename)  
*It sets the input audio file.*
- void [setFormat](#) (int fs, int samplesize)  
*It sets audio output format.*
- void [setDuration](#) (QLabel \*label, double duration)  
*It sets a duration indicator text on an user interface label object.*
- void [getDuration](#) (QLabel label)
- void [setUpmixType](#) ([UpmixType::upmixtype](#) upmixtype)  
*It sets SAC parameter upmix type.*
- void [setDecodingType](#) ([DecodingType::decodingtype](#) decodingtype)  
*It sets SAC parameter decoding type.*
- void [setBinauralQuality](#) ([BinauralQuality::binauralquality](#) binauralquality)  
*It sets SAC parameter binaural quality.*
- void [setHRTFModel](#) ([HRTFModel::hrtfmodel](#) hrtfmodel)  
*It sets SAC parameter HRTF model.*

## Public Attributes

- const int fs = 44100

### 6.29.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 76 of file SACEffects.h.

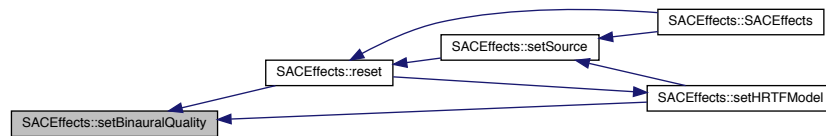


## Parameters

<i>binauralquality</i>	binaural quality
------------------------	------------------

Definition at line 401 of file SACEffects.cpp.

Here is the caller graph for this function:



## 6.29.3.2 setBitstream()

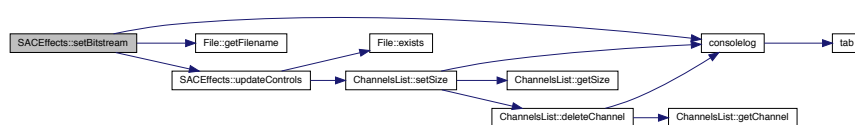
```
void SACEffects::setBitstream (
    std::string filename )
```

## Parameters

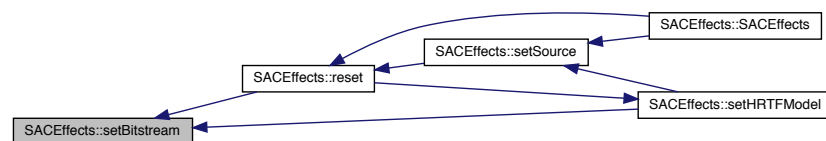
<i>filename</i>	file path
-----------------	-----------

Definition at line 233 of file SACEffects.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.29.3.3 setDecodingType()

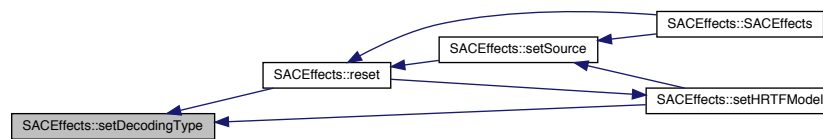
```
void SACEffects::setDecodingType (
    DecodingType::decodingtype decodingtype )
```

#### Parameters

<i>decodingtype</i>	decoding type
---------------------	---------------

Definition at line 380 of file SACEffects.cpp.

Here is the caller graph for this function:



### 6.29.3.4 setDuration()

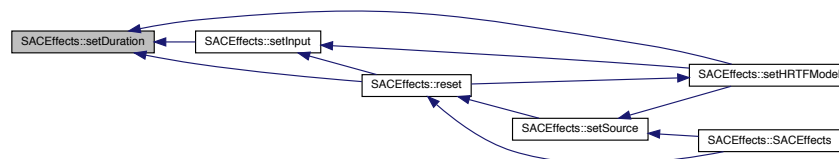
```
void SACEffects::setDuration (
    QLabel * label,
    double duration )
```

#### Parameters

<i>label</i>	user interface object where to indicate duration
<i>duration</i>	input audio file duration [s]

Definition at line 321 of file SACEffects.cpp.

Here is the caller graph for this function:



## 6.29.3.5 setEffect()

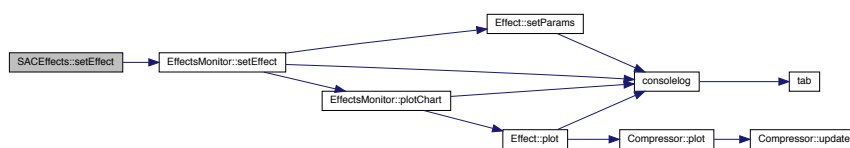
```
void SACEffects::setEffect (
    Effect::effectID effect )
```

## Parameters

<i>effect</i>	selected effect
---------------	-----------------

Definition at line 182 of file SACEffects.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.29.3.6 setFormat()

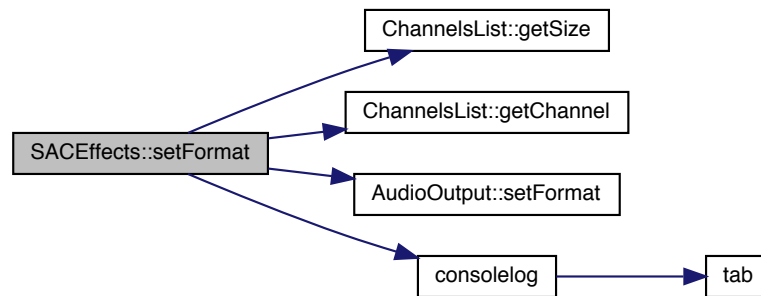
```
void SACEffects::setFormat (
    int fs,
    int samplesize )
```

## Parameters

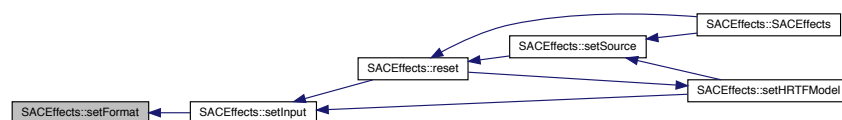
<i>fs</i>	signal sampling frequency
<i>samplesize</i>	signal sample size

Definition at line 307 of file SACEffects.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



### 6.29.3.7 setHRTFModel()

```
void SACEffects::setHRTFModel (
    HRTFModel::hrtfmodel hrtfmodel )
```

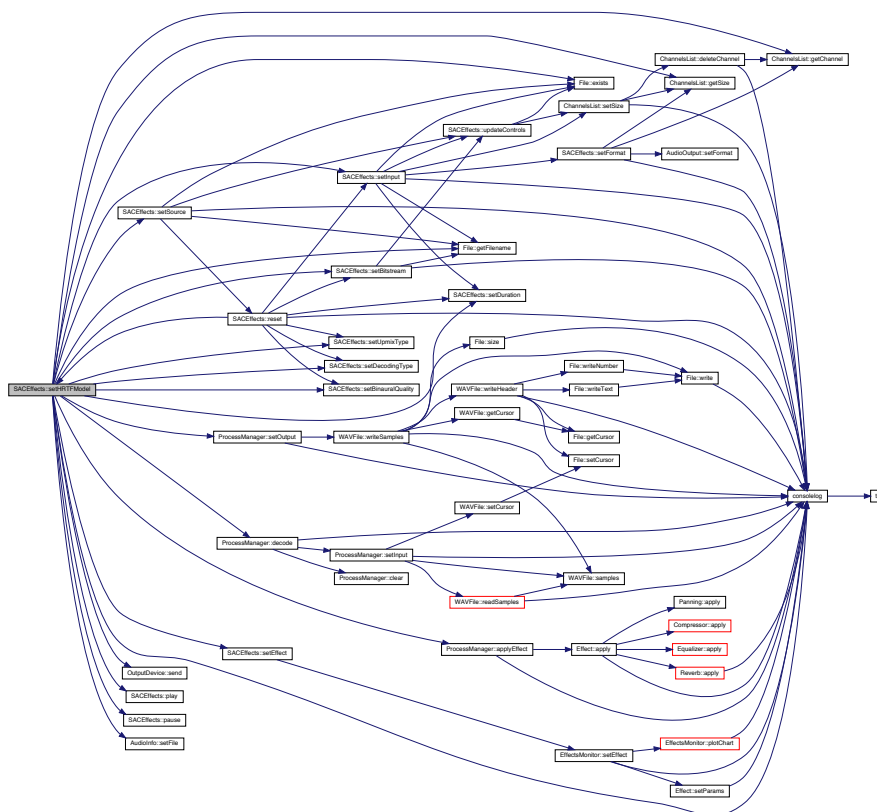
#### Parameters

<i>hrtfmodel</i>	HRTF model
------------------	------------

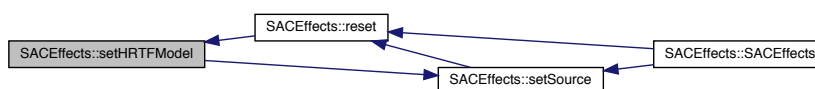
Definition at line 423 of file `SACEffects.cpp`.



Here is the call graph for this function:



Here is the caller graph for this function:



### 6.29.3.8 setInput()

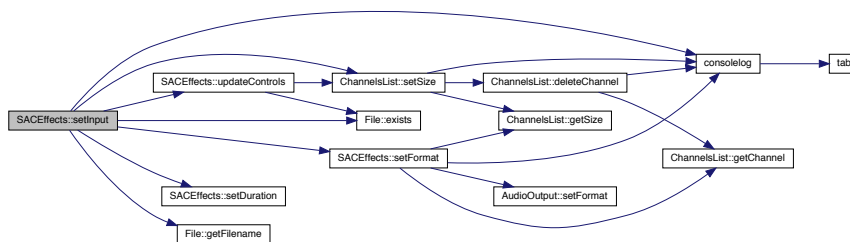
```
void SACEffects::setInput (
    std::string filename )
```

### Parameters

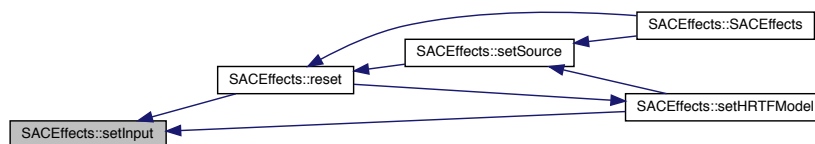
<i>filename</i>	file path
-----------------	-----------

Definition at line 266 of file SACEffects.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.29.3.9 setSource()

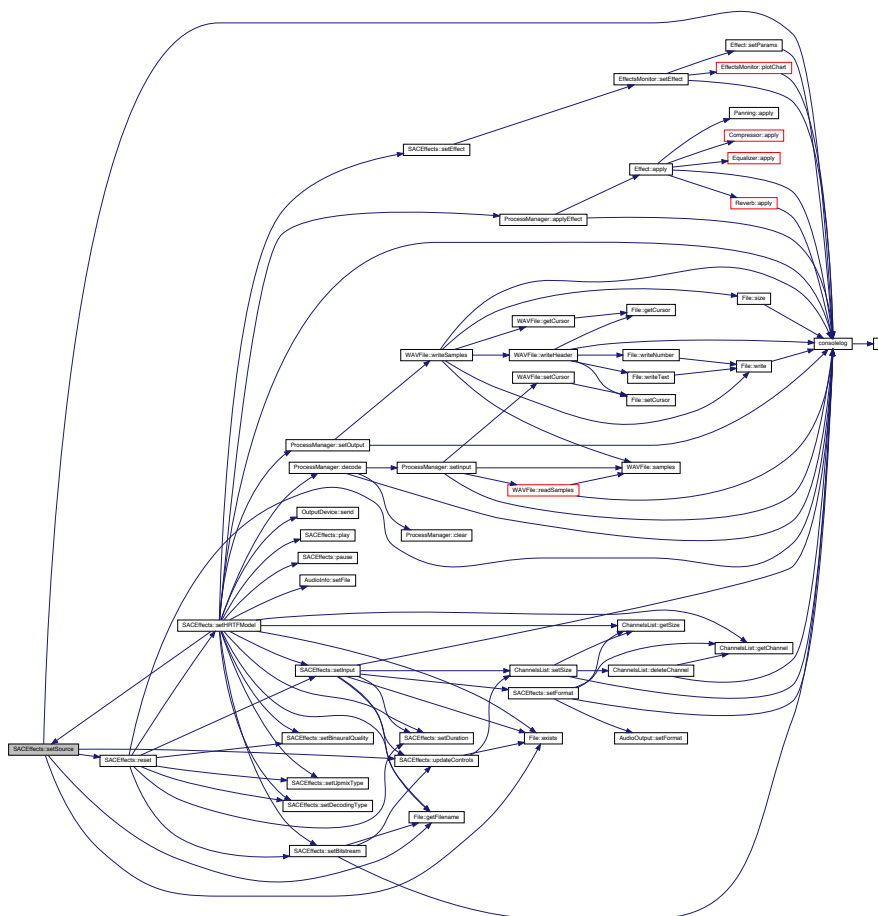
```
void SACEffects::setSource (
    std::string filename )
```

##### Parameters

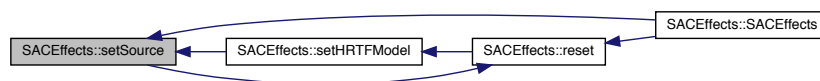
<i>filename</i>	file path
-----------------	-----------

Definition at line 202 of file SACEffects.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.29.3.10 setUpmixType()

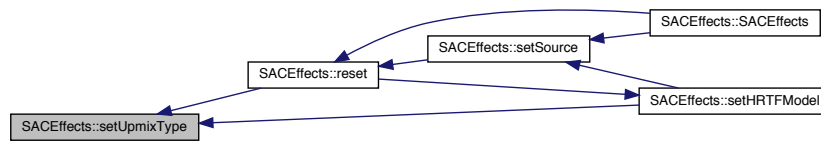
```
void SACEffects::setUpmixType (
    UpmixType::upmixtype upmixtype )
```

### Parameters

<i>upmixtype</i>	upmix type
------------------	------------

Definition at line 353 of file SACEffects.cpp.

Here is the caller graph for this function:



## 6.29.4 Member Data Documentation

### 6.29.4.1 fs

```
const int SACEffects::fs = 44100
```

signal sampling frequency [Hz]

Definition at line 79 of file SACEffects.h.

The documentation for this class was generated from the following files:

- `src/interface/SACEffects.h`
- `src/interface/SACEffects.cpp`

## 6.30 UpmixType Struct Reference

SAC decoder parameter upmix type.

```
#include <SACEffects.h>
```

### Public Types

- enum `upmixtype` { `normal` = 0, `blind` = 1, `binaural` = 2, `stereo` = 3 }

### 6.30.1 Detailed Description

Definition at line 26 of file SACEffects.h.

### 6.30.2 Member Enumeration Documentation

#### 6.30.2.1 upmixtype

```
enum UpmixType::upmixtype
```

## Enumerator

normal	normal upmix
blind	blind upmix
binaural	binaural upmix
stereo	stereo upmix

Definition at line 27 of file SACEffects.h.

The documentation for this struct was generated from the following file:

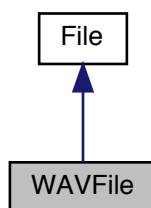
- src/interface/SACEffects.h

## 6.31 WAVFile Class Reference

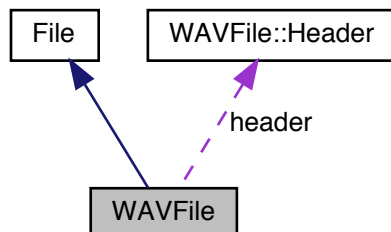
Audio file as WAV format class.

```
#include <File.h>
```

Inheritance diagram for WAVFile:



Collaboration diagram for WAVFile:



## Classes

- struct [Header](#)  
*Audio file header struct.*

## Public Member Functions

- [WAVFile](#) (bool writepermission)  
*WAVFile constructor.*
- [WAVFile](#) (std::string filename, bool writepermission)  
*WAVFile constructor.*
- [WAVFile](#) (std::string filename, int channels, int fs, int sampleformat)  
*WAVFile constructor. Write file is allowed.*
- [~WAVFile](#) ()  
*WAVFile destructor.*
- void [setCursor](#) (int cursor)  
*It sets the signal reading cursor to keep on reading from another position.*
- int [getCursor](#) ()  
*It gets the current signal reading cursor.*
- int [samples](#) ()  
*It gets the number of audio samples.*
- void [readHeader](#) ()  
*It reads the file header and sets the format header into the audio file object.*
- void [writeHeader](#) ()  
*It writes the header on the file from the audio file object header.*
- float \*\* [readSamples](#) (int [samples](#))  
*It reads an array of samples from the audio file.*
- void [writeSamples](#) (float \*\*array, int [samples](#))  
*It writes an array of samples on the audio file.*

## Public Attributes

- [Header](#) header
- double [duration](#)

### 6.31.1 Detailed Description

#### Author

Andrés González Fornell

Definition at line 57 of file File.h.

### 6.31.2 Constructor & Destructor Documentation

#### 6.31.2.1 [WAVFile\(\)](#) [1/3]

```
WAVFile::WAVFile (
    bool writepermission )
```

## Parameters

<i>writepermission</i>	file write permission (true if it is allowed)
------------------------	---

Definition at line 220 of file File.cpp.

## 6.31.2.2 WAVFile() [2/3]

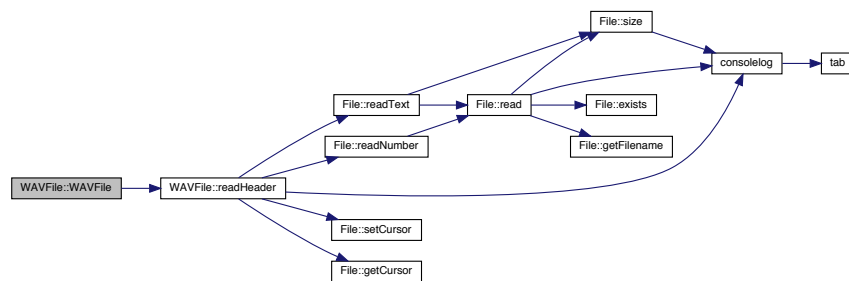
```
WAVFile::WAVFile (
    std::string filename,
    bool writepermission )
```

## Parameters

<i>filename</i>	file path
<i>writepermission</i>	file write permission (true if it is allowed)

Definition at line 229 of file File.cpp.

Here is the call graph for this function:



## 6.31.2.3 WAVFile() [3/3]

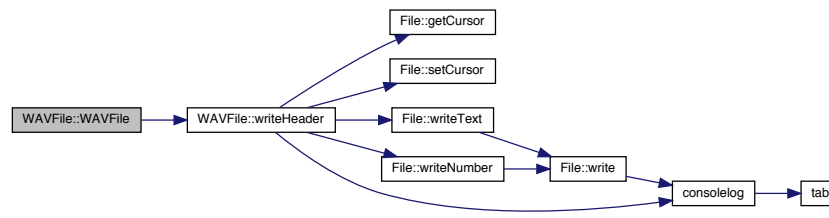
```
WAVFile::WAVFile (
    std::string filename,
    int channels,
    int fs,
    int sampleformat )
```

## Parameters

<i>filename</i>	file path
<i>channels</i>	number of channels
<i>fs</i>	signal sample rate
<i>sampleformat</i>	number of bits of a sample

Definition at line 241 of file File.cpp.

Here is the call graph for this function:



### 6.31.3 Member Function Documentation

#### 6.31.3.1 getCursor()

```
int WAVFile::getCursor ( )
```

##### Returns

cursor [Bytes] from the beginning of the signal (instead of the file)

Definition at line 279 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.31.3.2 readSamples()

```
float ** WAVFile::readSamples (
    int samples )
```



## Parameters

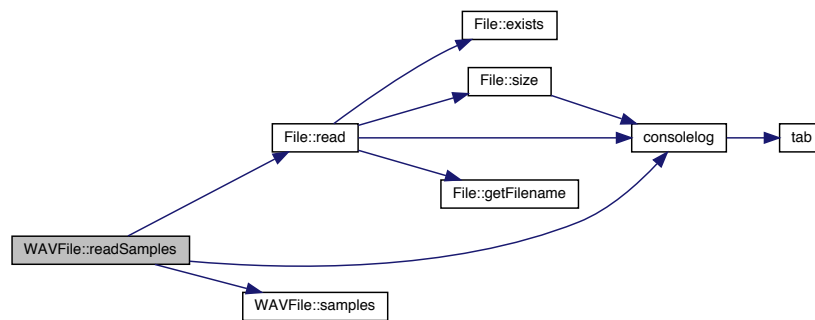
<i>samples</i>	number of samples
----------------	-------------------

## Returns

two dimensional array ([channel][sample]) of samples (from -1 to 1)

Definition at line 424 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.31.3.3 samples()

```
int WAVFile::samples ( )
```

## Returns

number of audio samples

Definition at line 288 of file File.cpp.

Here is the caller graph for this function:



#### 6.31.3.4 setCursor()

```
void WAVFile::setCursor (
    int cursor )
```

##### Parameters

<i>cursor</i>	new cursor position in samples (instead of bytes) from the beginning of the signal (instead of the file)
---------------	--

Definition at line 269 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



#### 6.31.3.5 writeSamples()

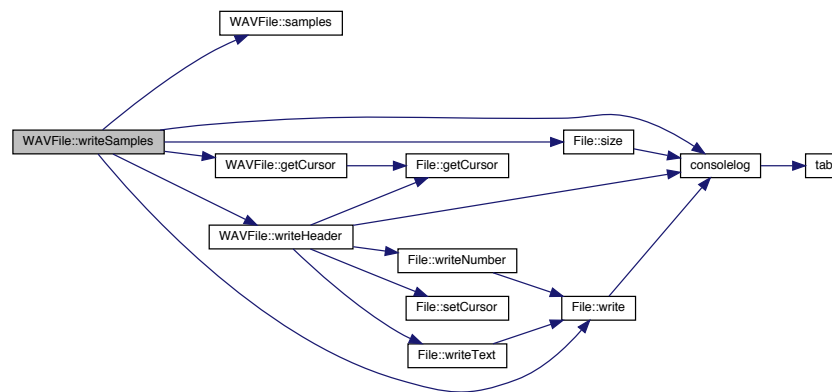
```
void WAVFile::writeSamples (
    float ** array,
    int samples )
```

##### Parameters

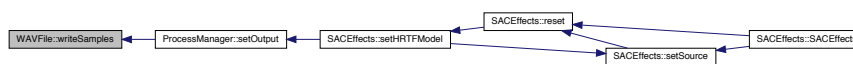
<i>array</i>	two dimensional array ([channel][sample]) of samples (from -1 to 1)
<i>samples</i>	number of samples

Definition at line 467 of file File.cpp.

Here is the call graph for this function:



Here is the caller graph for this function:



## 6.31.4 Member Data Documentation

### 6.31.4.1 duration

```
double WAVFile::duration
```

audio file duration [s]

Definition at line 99 of file File.h.

### 6.31.4.2 header

```
Header WAVFile::header
```

audio file header

Definition at line 98 of file File.h.

The documentation for this class was generated from the following files:

- src/process/File.h
- src/process/File.cpp



## Chapter 7

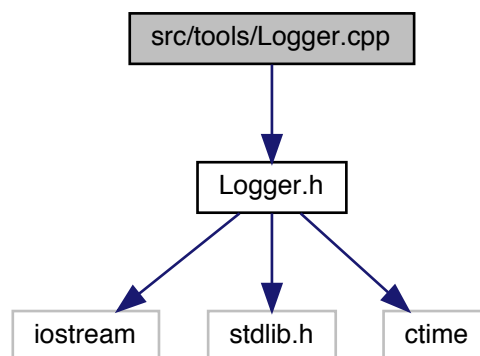
# File Documentation

### 7.1 src/tools/Logger.cpp File Reference

Functions to create log messages on console.

```
#include "Logger.h"
```

Include dependency graph for Logger.cpp:



### Functions

- `std::string tab (std::string content, const int tab_max)`  
*It returns the string tab code to align log messages.*
- `void consolelog (std::string source, LogType::logtype logtype, std::string message)`  
*Log a message on console.*

## Variables

### Font styles

*ANSI code for some font styles for log messages usage.*

- `const std::string reset = "\033[0m"`
- `const std::string bold = "\033[1m"`
- `const std::string italic = "\033[3m"`
- `const std::string black = "\033[30m"`
- `const std::string red = "\033[31m"`
- `const std::string green = "\033[32m"`
- `const std::string yellow = "\033[33m"`
- `const std::string blue = "\033[34m"`
- `const std::string magenta = "\033[35m"`
- `const std::string cyan = "\033[36m"`
- `const std::string grey = "\033[37m"`

### 7.1.1 Detailed Description

#### Author

Andrés González Fornell

### 7.1.2 Function Documentation

#### 7.1.2.1 `consolelog()`

```
void consolelog (
    std::string source,
    LogType::logtype logtype,
    std::string message )
```

#### Parameters

<i>source</i>	origin class/method/file where the message was logged
<i>logtype</i>	type of message
<i>message</i>	message

#### Returns

`void`

Definition at line 51 of file `Logger.cpp`.

Here is the call graph for this function:



#### 7.1.2.2 tab()

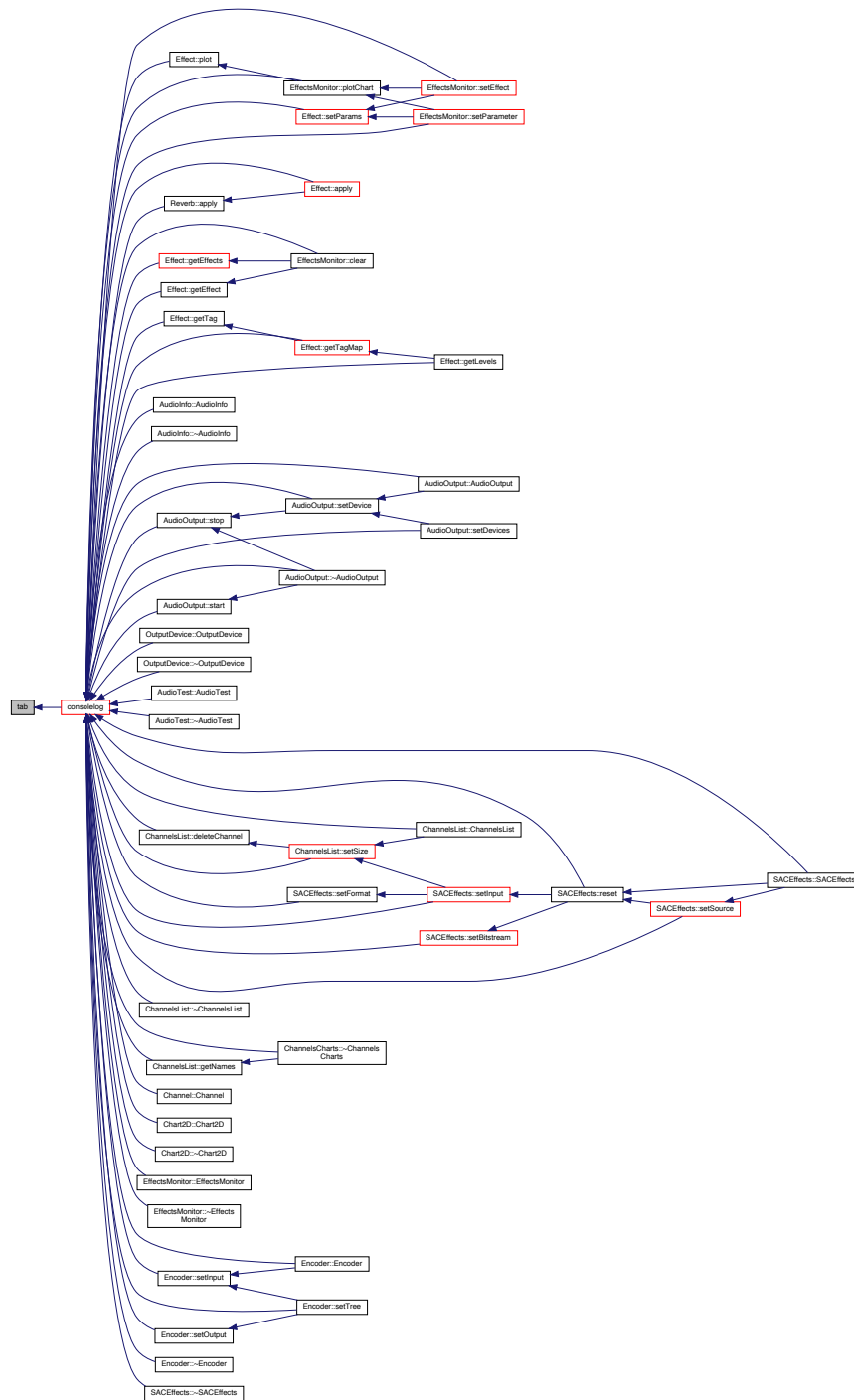
```
std::string tab (  
    std::string content,  
    const int tab_max )
```

##### Parameters

<i>content</i>	Content of the tabulation
<i>tab_max</i>	Maximum number of tabulations

Definition at line 33 of file Logger.cpp.

Here is the caller graph for this function:



### 7.1.3 Variable Documentation

#### 7.1.3.1 black

```
const std::string black = "\033[30m"
```



black color font

Definition at line 18 of file Logger.cpp.

#### 7.1.3.2 blue

```
const std::string blue = "\033[34m"
```

blue color font

Definition at line 22 of file Logger.cpp.

#### 7.1.3.3 bold

```
const std::string bold = "\033[1m"
```

bold

Definition at line 16 of file Logger.cpp.

#### 7.1.3.4 cyan

```
const std::string cyan = "\033[36m"
```

cyan color font

Definition at line 24 of file Logger.cpp.

#### 7.1.3.5 green

```
const std::string green = "\033[32m"
```

green color font

Definition at line 20 of file Logger.cpp.

#### 7.1.3.6 grey

```
const std::string grey = "\033[37m"
```

grey color font

Definition at line 25 of file Logger.cpp.

#### 7.1.3.7 italic

```
const std::string italic = "\033[3m"
```

italic

Definition at line 17 of file Logger.cpp.

#### 7.1.3.8 magenta

```
const std::string magenta = "\033[35m"
```

magenta color font

Definition at line 23 of file Logger.cpp.

#### 7.1.3.9 red

```
const std::string red = "\033[31m"
```

red color font

Definition at line 19 of file Logger.cpp.

#### 7.1.3.10 reset

```
const std::string reset = "\033[0m"
```

default style

Definition at line 15 of file Logger.cpp.

#### 7.1.3.11 yellow

```
const std::string yellow = "\033[33m"
```

yellow color font

Definition at line 21 of file Logger.cpp.