SAC Effects

v1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

Namespace Index

1.1 Namespace List

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

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

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

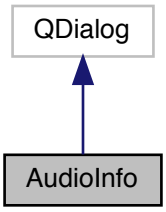
Class Documentation

6.1 AudioInfo Class Reference

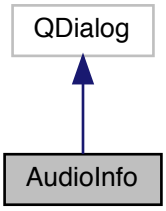
Audio object info dialog class.

`#include <AudioInfo.h>`

Inheritance diagram for AudioInfo:



Collaboration diagram for AudioInfo:



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()

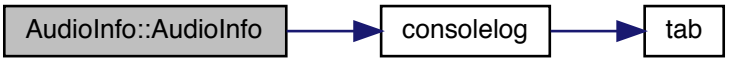
```
AudioInfo::AudioInfo (
            QWidget * parent = 0 )
```

Parameters

parent	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()

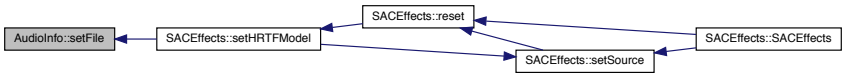
```
void AudioInfo::setFile (
            WAVFile * file )
```

Parameters

file	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:

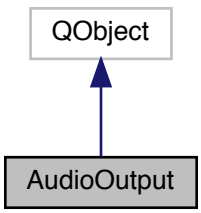
- src/interface/AudioInfo.h
- src/interface/AudioInfo.cpp

6.2 AudioOutput Class Reference

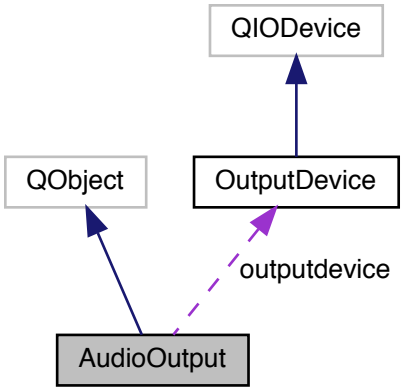
Audio output class.

```
#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)
  AudioOuput 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()

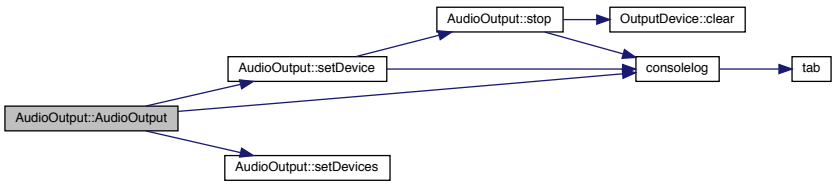
```
AudioOutput::AudioOutput (
            QComboBox * selector,
            int fs,
            int samplesize )
```

Parameters

selector	user interface combo box to select audio device
fs	signal sampling frequency
samplesize	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

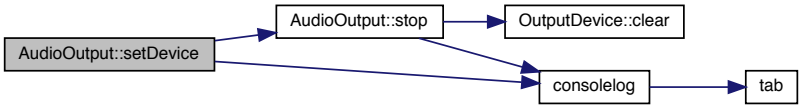
```
void AudioOutput::setDevice (
            int index )  [slot]
```

Parameters

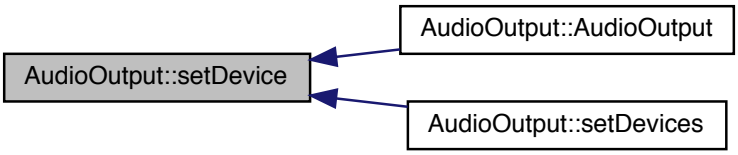
index	device index

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()

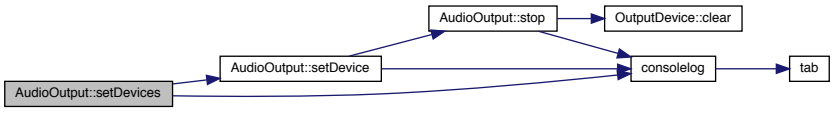
```
void AudioOutput::setDevices (
            QList< QAudioDeviceInfo > devices )
```

Parameters

devices	

Definition at line 100 of file AudioOutput.cpp.

Here is the call graph for this function:



6.2.3.3 setFormat()

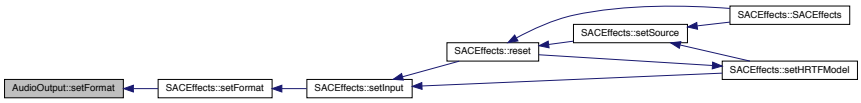
```
void AudioOutput::setFormat (
            int fs,
            int samplesize )
```

Parameters

fs	signal sampling frequency.
samplesize	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

volume	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 beggining at time t = 0 s.
- std::vector< float > getTimes (float initialtime)
  It gets time [s] axis as a vector beggining 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 = 0xFFFFFF

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

fs	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

signal	vector of signal samples
fs	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

sample	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

index	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

start	first index of the range (included)
end	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

bands	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

start	first index of the range (included)
end	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

bands	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

initialtime	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

index	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

index	selected index
sample	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

signal	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 = 0xFFFFFF  [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:

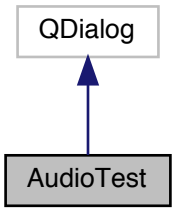
- src/process/AudioSignal.h
- src/process/AudioSignal.cpp

6.4 AudioTest Class Reference

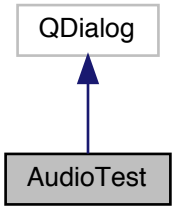
Audio output test class.

`#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()

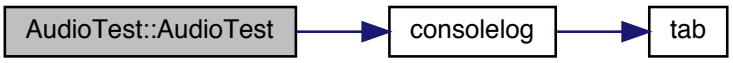
```
AudioTest::AudioTest (
            QWidget * parent = 0 )
```

Parameters

parent	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:

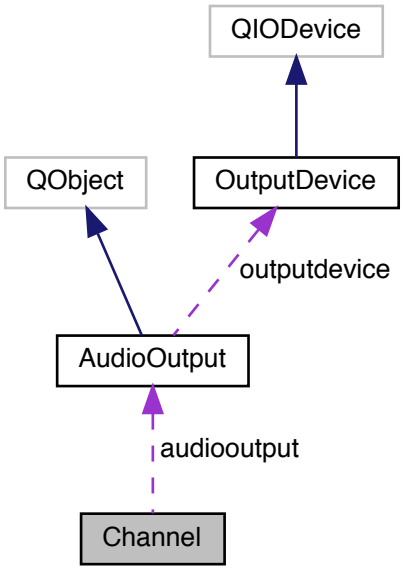
- src/interface/SACEffects.h

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()

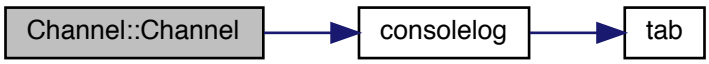
```
Channel::Channel (
            QLayout * framework,
            std::string prefix,
            int index,
            bool isoutput )
```

Parameters

framework	channel user interface framework
prefix	prefix of objects name of channel user interface
index	channel index
isoutput	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

state	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

state	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

label	new label

Definition at line 297 of file ChannelsList.cpp.

6.6.3.5 setVolume()

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

Parameters

volume	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

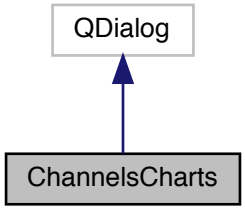
Definition at line 46 of file ChannelsList.h.

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

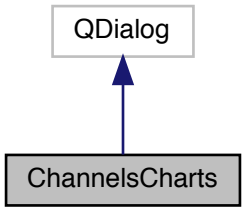
- src/interface/ChannelsList.h
- src/interface/ChannelsList.cpp

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

input	input signal pointer
output	output signal pointer
input_channels	input channels object
output_channels	output channels object
samples	number of samples each channel
parent	user inteface parent object

Definition at line 344 of file ChannelsList.cpp.

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

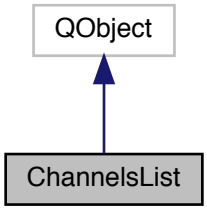
- src/interface/ChannelsList.h
- src/interface/ChannelsList.cpp

6.8 ChannelsList Class Reference

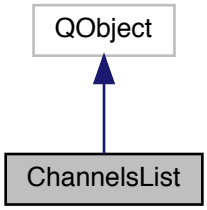
Channels list class. It shows information about channels signals.

```
#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()

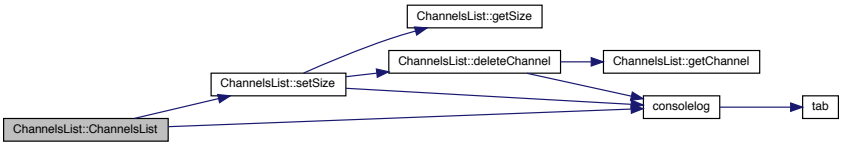
```
ChannelsList::ChannelsList (
            QWidget * framework,
            int number,
            bool showdevices )
```

Parameters

framework	user interface framework of channels list
number	number of channels
showdevices	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()

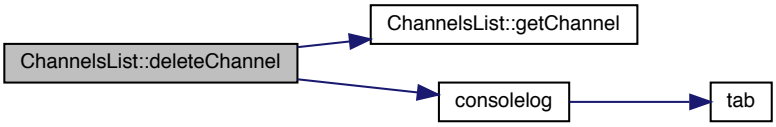
```
void ChannelsList::deleteChannel (
            int index )
```

Parameters

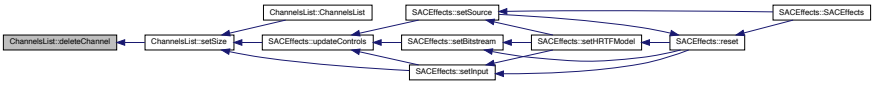
index	channel index

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

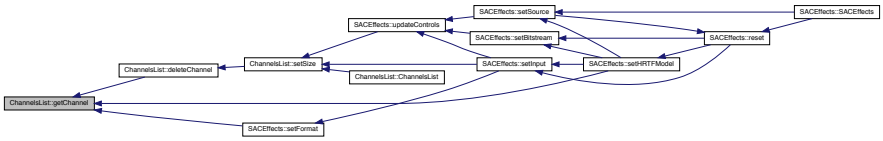
index	channel index

Returns

channel pointer

Definition at line 32 of file ChannelsList.cpp.

Here is the caller graph for this function:



6.8.3.3 getNames()

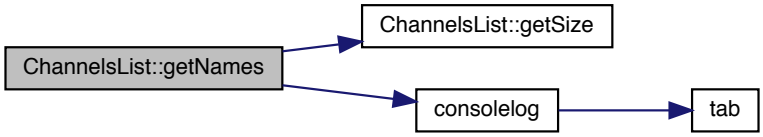
```
std::vector< std::string > ChannelsList::getNames ( )
```

Returns

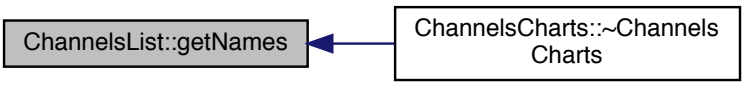
list of channels names

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()

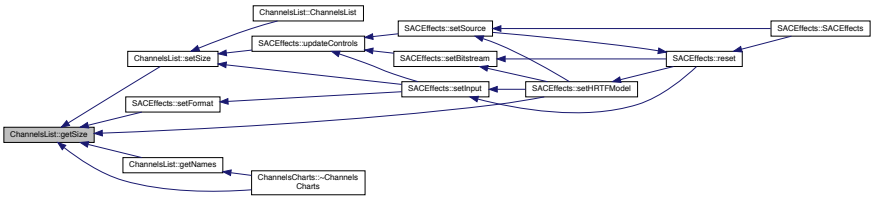
```
int ChannelsList::getSize ( )
```

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()

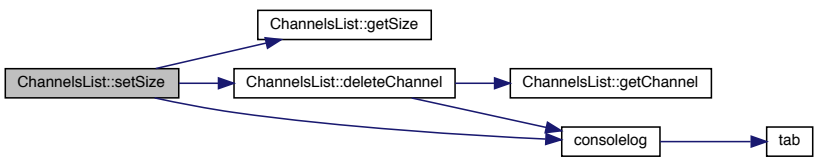
```
void ChannelsList::setSize (
            int size )
```

Parameters

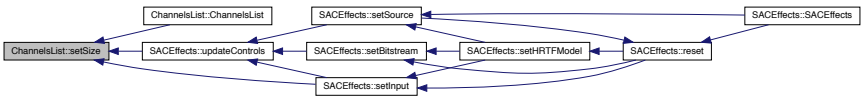
size	number of channels

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:

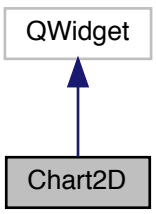
- src/sac/SACBitstream.h

6.10 Chart2D Class Reference

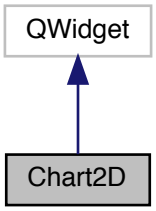
Class for plotting two-dimensional charts.

`#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]

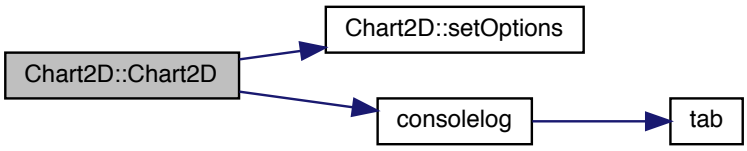
```
Chart2D::Chart2D (
            QWidget * framework )
```

Parameters

framework	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]

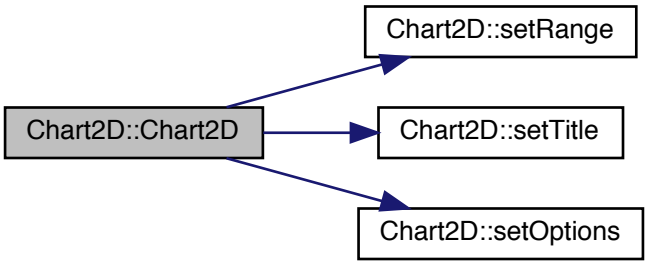
```
Chart2D::Chart2D (
            QWidget * framework,
            double range[2][2],
            std::string title,
            std::string xlabel,
            std::string ylabel,
            int options )
```

Parameters

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

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()

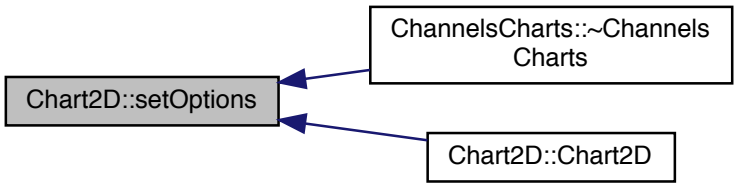
```
void Chart2D::setOptions (
            int options )
```

Parameters

options	ChartOptions

Definition at line 86 of file Chart2D.cpp.

Here is the caller graph for this function:



6.10.3.3 setPoints()

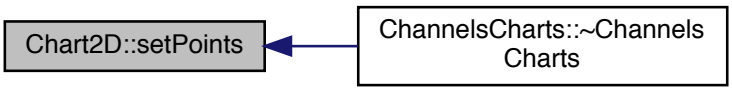
```
void Chart2D::setPoints (
            QVector< QPointF > points )
```

Parameters

points	new chart points

Definition at line 52 of file Chart2D.cpp.

Here is the caller graph for this function:



6.10.3.4 setRange()

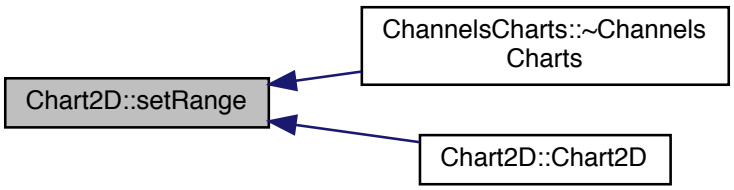
```
void Chart2D::setRange (
            double range[2][2] )
```

Parameters

range	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

title	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

input	input signal pointer
output	output signal pointer
samples	number of samples
channels	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

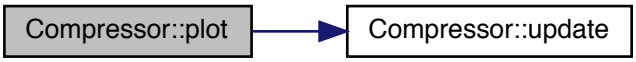
chart	chart id

Returns

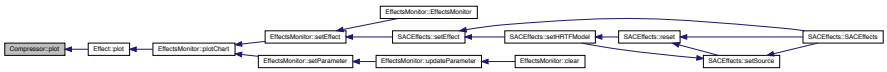
array of values as values[axis][sample] axis: 0 = x (horizontal) and 1 = y (vertical)

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]

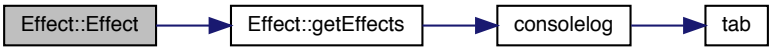
```
Effect::Effect (
            Effect::effectID effect,
            int fs )
```

Parameters

effect	effect ID
fs	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]

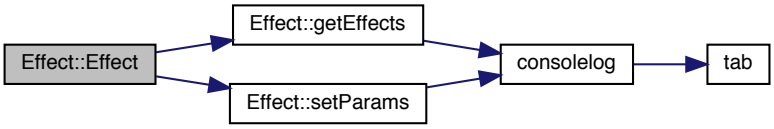
```
Effect::Effect (
            Effect::effectID effect,
            std::map< std::string, std::string > params,
            int fs )
```

Parameters

effect	effect ID
params	map of effect parameters
fs	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

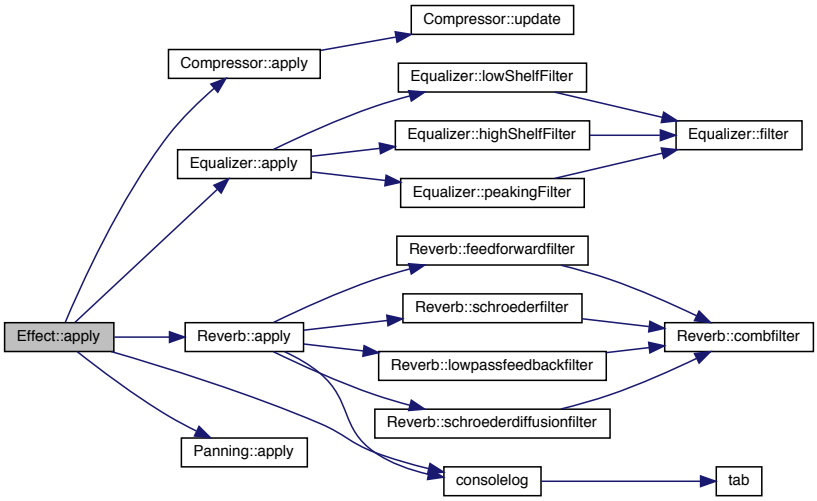
input	input data pointer
output	output data pointer
samples	number of samples
channels	vector of channel types

Returns

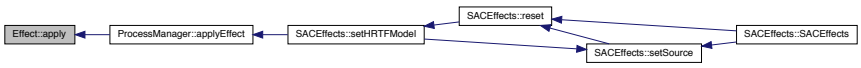
true if it was successful

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

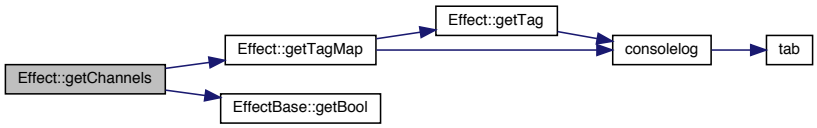
configuration	contained text of a effect configuration file (.fx)
size	number of channels

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

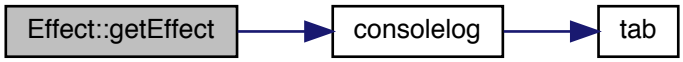
effectname	effect name string

Returns

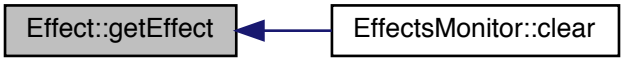
effect type effectID

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()

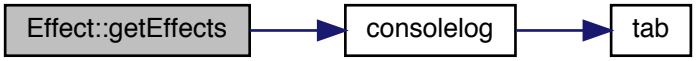
```
std::map< Effect::effectID, std::string > Effect::getEffects ( )  [static]
```

Returns

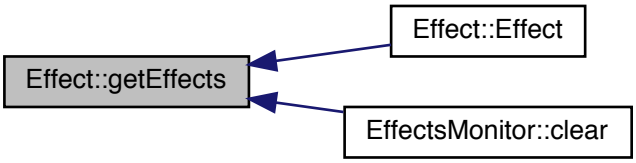
map of available effects

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

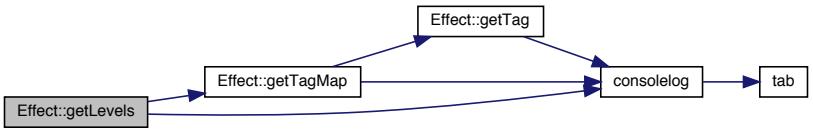
configuration	contained text of a effect configuration file (.fx)
size	number of channels

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

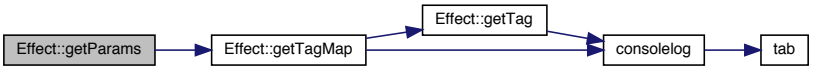
configuration	contained text of a effect configuration file (.fx)

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

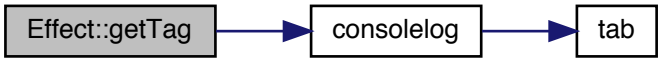
configuration	contained text of a effect configuration file (.fx)
tag	tag name of the requested field

Returns

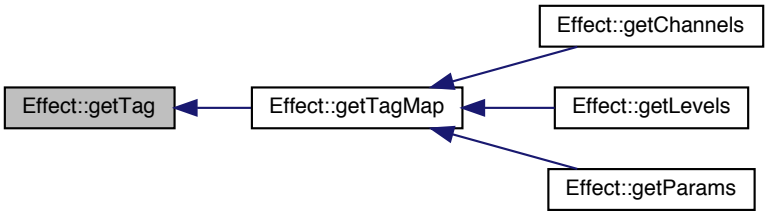
contained value in the tag

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

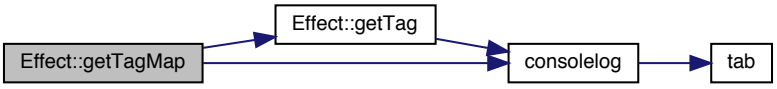
configuration	contained text of a effect configuration file (.fx)
tag	tag name of the requested field

Returns

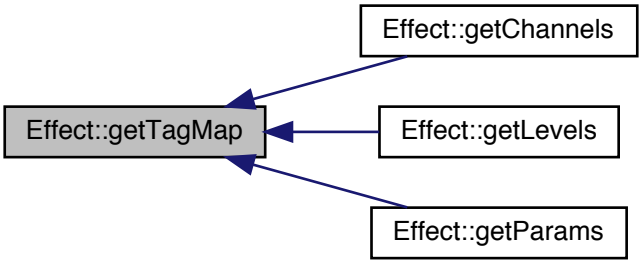
contained map of values in the tag

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

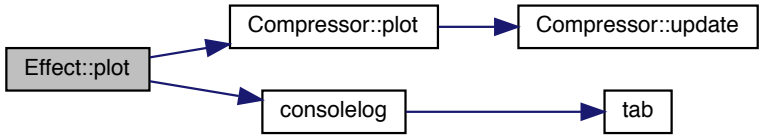
chart	chart id

Returns

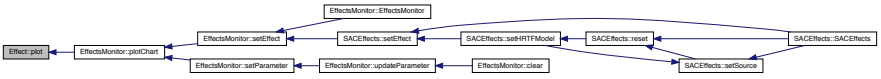
array of values as values[axis][sample] axis: 0 = x (horizontal) and 1 = y (vertical)

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()

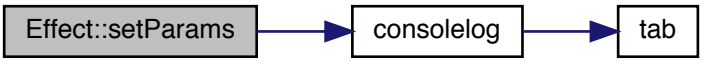
```
void Effect::setParams (
            std::map< std::string, std::string > params )
```

Parameters

params	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

param	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

param	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

param	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:

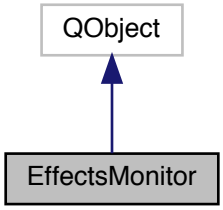
- src/effects/EffectBase.h
- src/effects/Effect.cpp

6.16 EffectsMonitor Class Reference

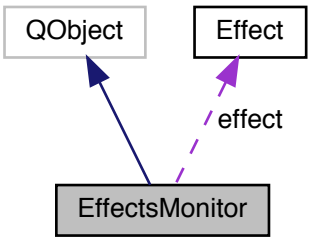
Class for managing effects parameters.

`#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]

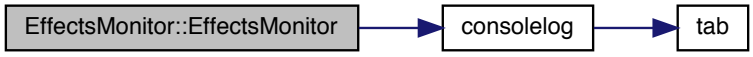
```
EffectsMonitor::EffectsMonitor (
            QWidget * framework )
```

Parameters

framework	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]

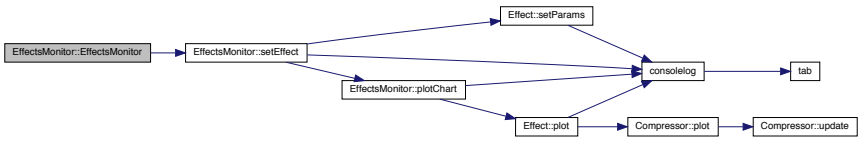
```
EffectsMonitor::EffectsMonitor (
            QWidget * framework,
            Effect * effect )
```

Parameters

framework	user interface framework
effect	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()

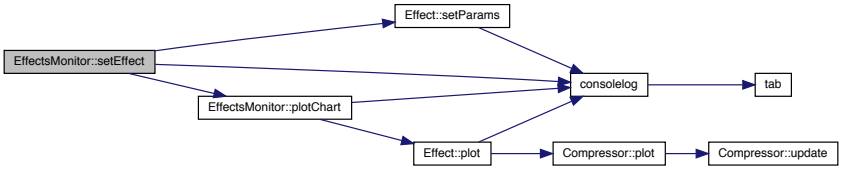
```
void EffectsMonitor::setEffect (
            Effect * effect )
```

Parameters

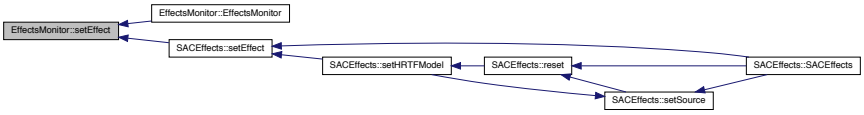
effect	selected effect

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()

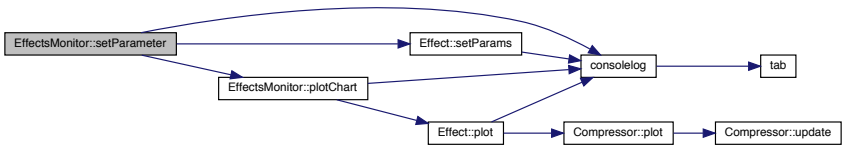
```
void EffectsMonitor::setParameter (
            std::string parameter,
            std::string value )
```

Parameters

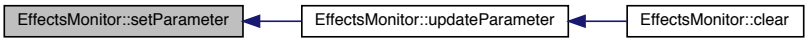
parameter	parameter name
value	new parameter value

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]

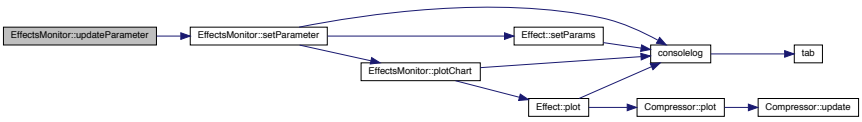
```
void EffectsMonitor::updateParameter (
            int value )  [slot]
```

Parameters

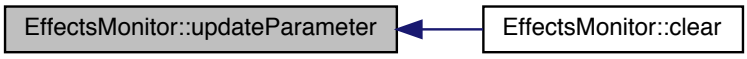
value	changed value

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]

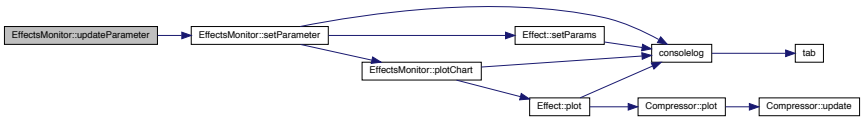
```
void EffectsMonitor::updateParameter (
            double value )  [slot]
```

Parameters

value	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]

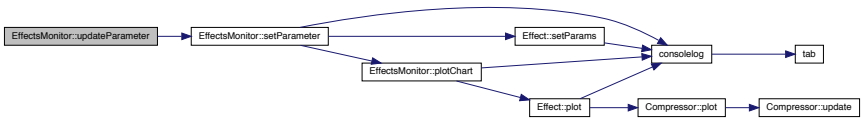
```
void EffectsMonitor::updateParameter (
            QString value )  [slot]
```

Parameters

value	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]

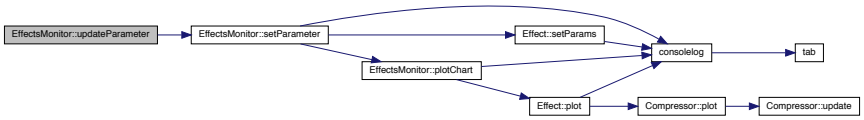
```
void EffectsMonitor::updateParameter (
            bool value )  [slot]
```

Parameters

value	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:

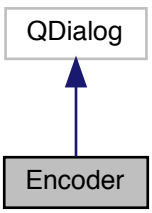
- src/interface/EffectsMonitor.h
- src/interface/EffectsMonitor.cpp

6.17 Encoder Class Reference

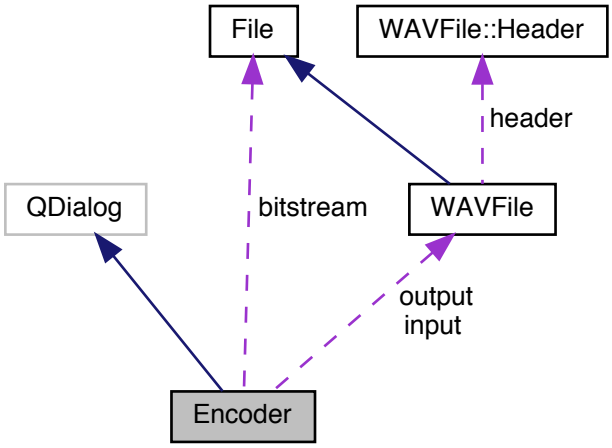
Encoder window interface.

```
#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()

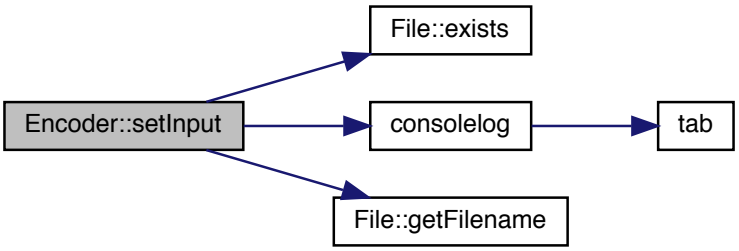
```
void Encoder::setInput (
            std::string filename )
```

Parameters

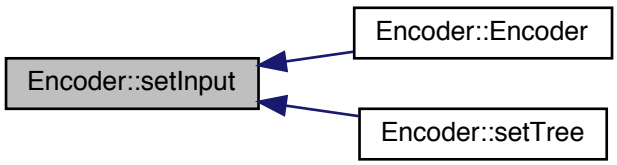
filename	file path

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()

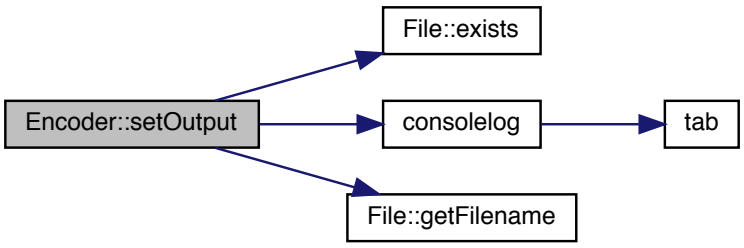
```
void Encoder::setOutput (
            std::string filename )
```

Parameters

filename	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()

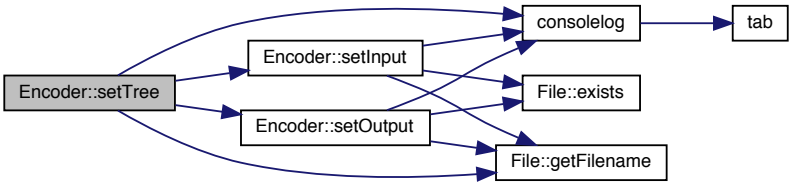
```
void Encoder::setTree (
            int tree )
```

Parameters

tree	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::Endianess Struct Reference

Public Types

- enum endianess { 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 endianess

```
enum File::Endianess::endianess
```

Enumerator

littleendian	little endian
bigendian	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]·z^-1 + ... + b[order]·z^-order) / (a[0] + a[1]·z^-1 + ... + a[order]·z^-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()

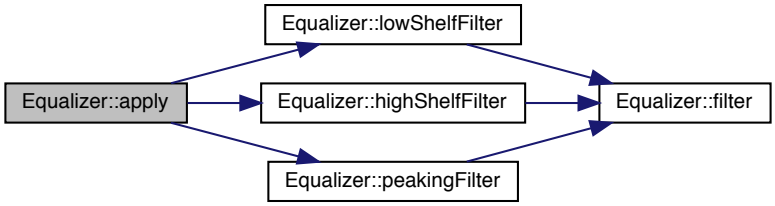
```
void Equalizer::apply (
            float ** input,
            float ** output,
            int samples,
            std::vector< SACBitstream::ChannelType::channeltype > channels )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
channels	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

input	input signal pointer
output	output signal pointer
samples	number of samples
a	y coefficients of transfer function
b	x coefficients of transfer function
order	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()

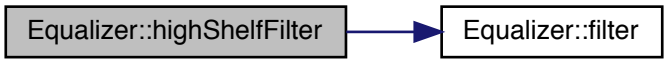
```
void Equalizer::highShelfFilter (
            float * input,
            float * output,
            int samples,
            double f_0,
            double gain,
            int order )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
f_0	midpoint frequency (real frequency / sampling frequency)
gain	peak power gain
order	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()

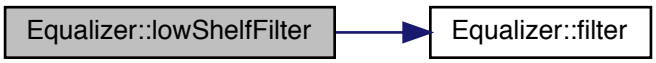
```
void Equalizer::lowShelfFilter (
            float * input,
            float * output,
            int samples,
            double f_0,
            double gain,
            int order )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
f_0	midpoint frequency (real frequency / sampling frequency)
gain	peak power gain
order	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()

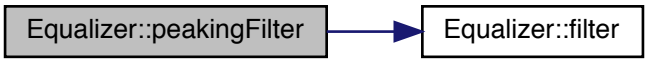
```
void Equalizer::peakingFilter (
            float * input,
            float * output,
            int samples,
            double f_0,
            double gain,
            double Q,
            int order )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
f_0	center frequency (real frequency / sampling frequency)
gain	peak power gain
Q	quality factor
order	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:

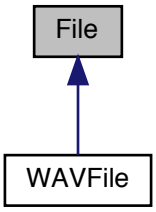
- src/effects/Equalizer.h
- src/effects/Equalizer.cpp

6.20 File Class Reference

Audio file class.

`#include <File.h>`

Inheritance diagram for File:



Classes

- struct Endianess

Endianess

Endianess 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, Endianess::endianess endianess)
  It reads a data number from the file.
- void writeNumber (unsigned int data, int length, Endianess::endianess endianess)
  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()

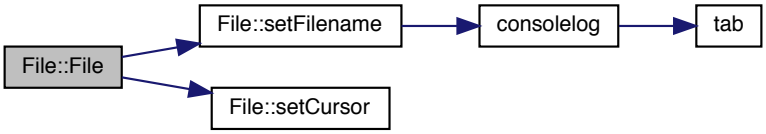
```
File::File (
            std::string filename,
            bool writepermission )
```

Parameters

filename	file path
writepermission	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()

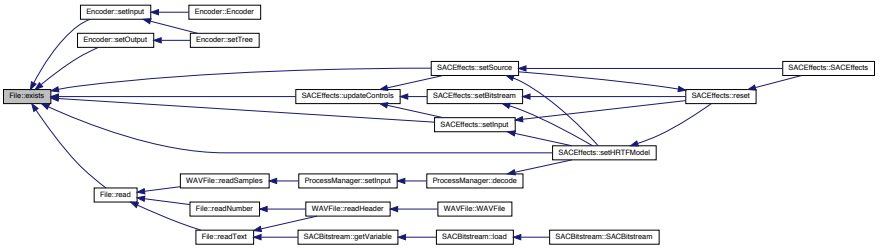
```
bool File::exists ( )
```

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()

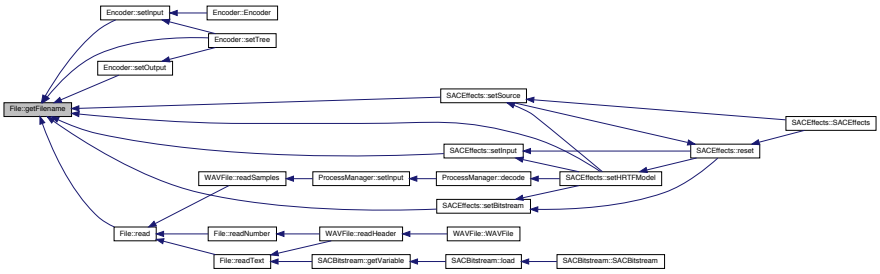
```
std::string File::getFilename ( )
```

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

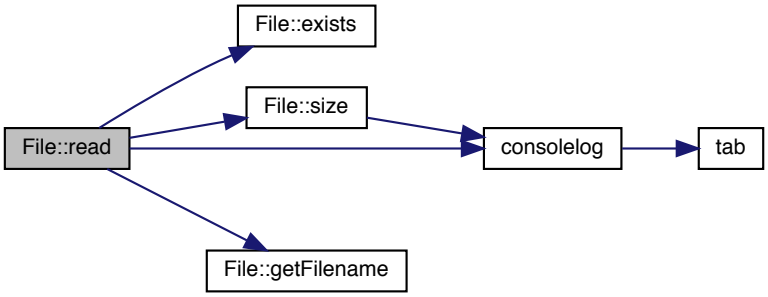
length	data length [Bytes]

Returns

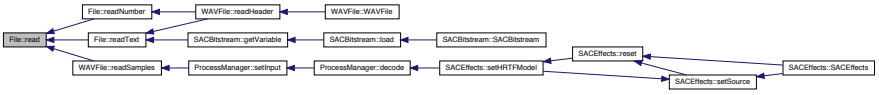
data pointer

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,
            Endianess::endianess endianess )
```

Parameters

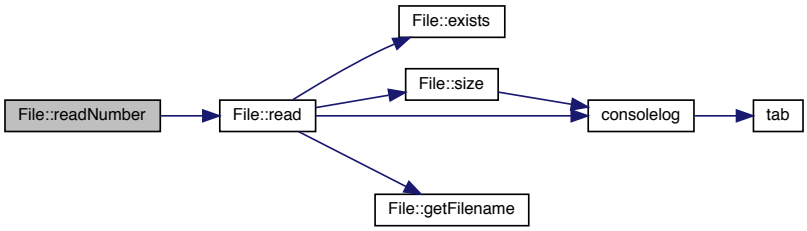
length	data length [Bytes]
endianess	data order (big endian or little endian)

Returns

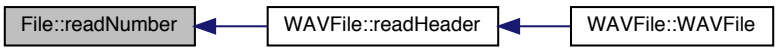
value of data number

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

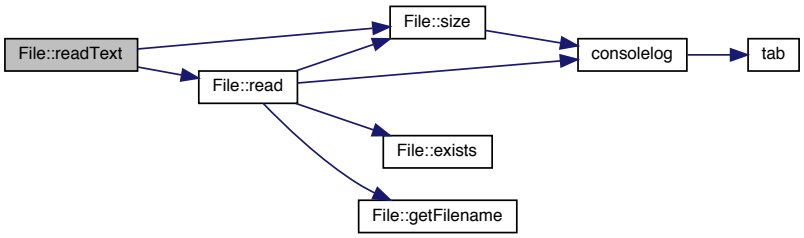
length	data length [Bytes] (if length = 0 function returns all available data from the file)

Returns

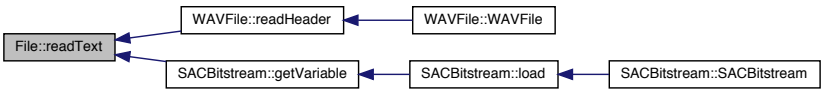
string of data

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()

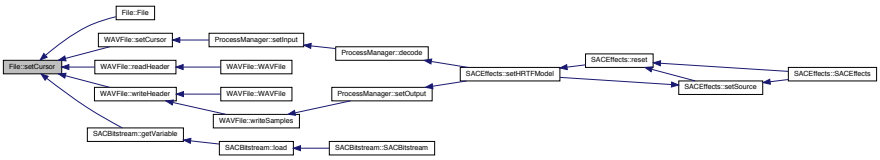
```
void File::setCursor (
            int cursor )
```

Parameters

cursor	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()

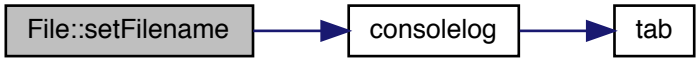
```
void File::setFilename (
            std::string filename )
```

Parameters

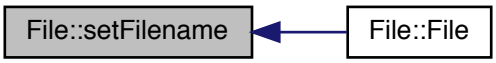
filename	file path name

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()

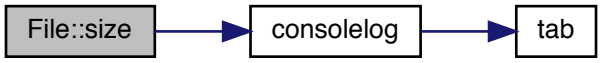
```
int File::size ( )
```

Returns

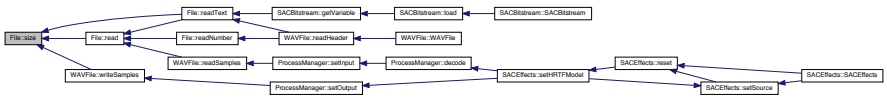
file size [Bytes]

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()

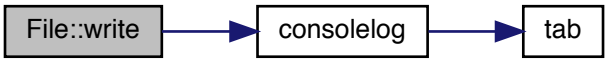
```
void File::write (
            const char * data,
            int length )
```

Parameters

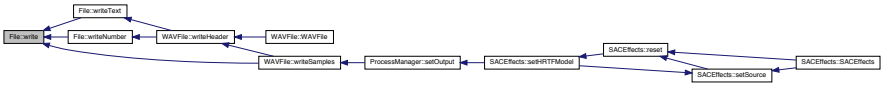
data	data pointer
length	data length [Bytes]

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,
            Endianess::endianess endianess )
```

Parameters

value	value of data number
length	data length [Bytes]
endianess	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()

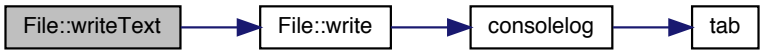
```
void File::writeText (
            std::string data )
```

Parameters

data	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 chunksize
- 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 funcion 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

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

Definition at line 14 of file Logger.h.

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

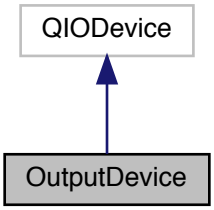
- src/tools/Logger.h

6.24 OutputDevice Class Reference

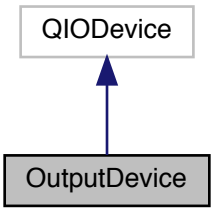
Audio output device class (QIODevice extension).

```
#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()

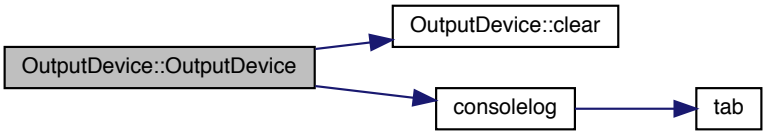
```
OutputDevice::OutputDevice (
            QAudioFormat format )
```

Parameters

format	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

data	data pointer
length	data length

Returns

Definition at line 228 of file AudioOutput.cpp.

6.24.3.3 send()

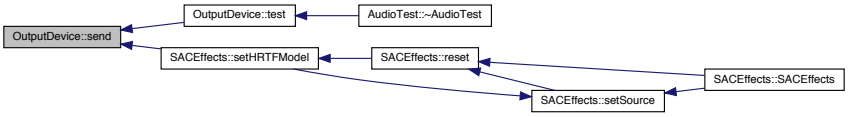
```
void OutputDevice::send (
            float * signal,
            int samples )
```

Parameters

signal	audio signal pointer
samples	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 )
```

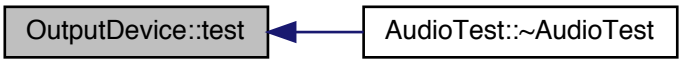
Parameters

amplitude	tone amplitude (from 0 to 1)
frequency	tone frequency [Hz]
duration	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

data	data pointer
length	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 pendient 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

input	input signal pointer
output	output signal pointer
samples	number of samples
channels	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 applys 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()

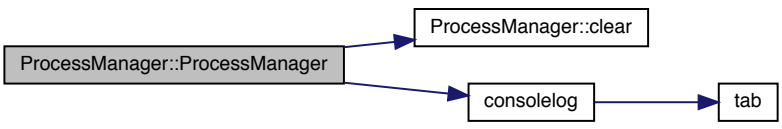
```
ProcessManager::ProcessManager (
            int chunksize )
```

Parameters

chunksize	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

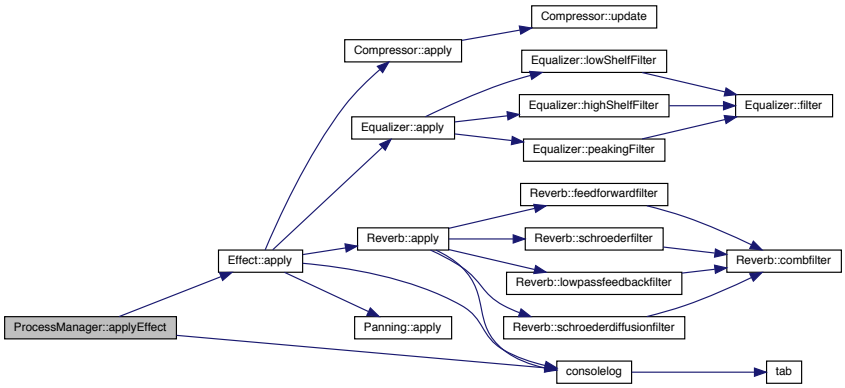
effect	effect object (it includes all parameters)
channels	boolean vector where true means to apply effect to that channel
levels	vector of input levels (>=0) for each channel

Returns

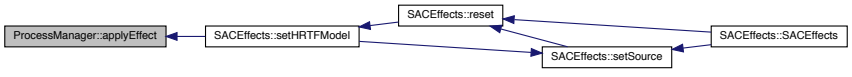
true if it was successful

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

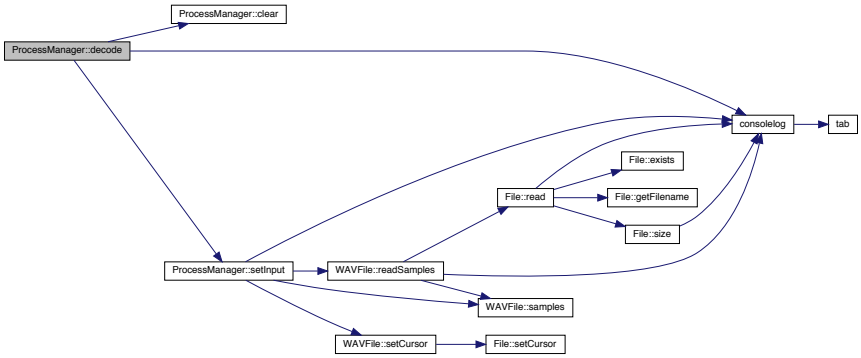
input	filename of the multichannel input audio file
output	filename of the downmix output audio file (it will be automatically created)
bitstream	filename of the bitstream output file or "buried" (it will be automatically created)
upmixtype	upmix type 0: normal 1: blind 2: binaural 3: stereo
decodingtype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

Returns

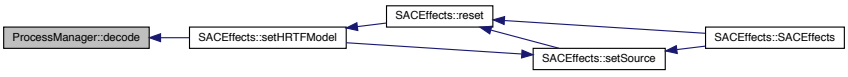
true if it was successful

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

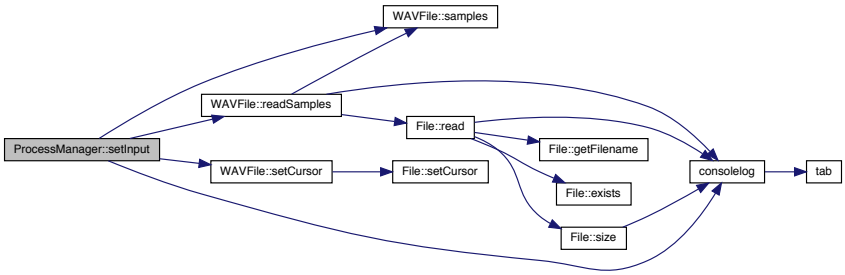
filename	audio input file name

Returns

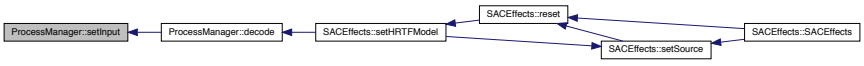
true if it was successful

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

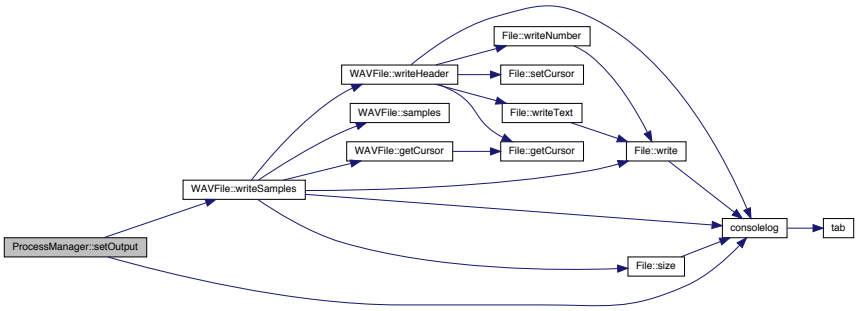
filename	audio output file name

Returns

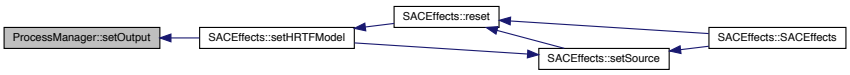
true if it was successful

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 shcroeder allpass filter, according to the transfer function H(z) = gain * (g + z^-delay) / (1 + g·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·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 · (1-d) / (1-d·z^-1) · 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]·z^-1 + ... + b[order]·z^-order + b_↩ delay·z^-delay) / (a[0] + a[1]·z^-1 + ... + a[order]·z^-order + a_delay·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()

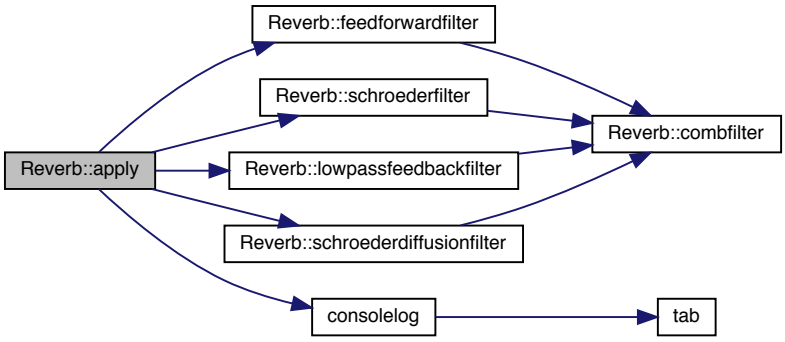
```
void Reverb::apply (
            float ** input,
            float ** output,
            int samples,
            std::vector< SACBitstream::ChannelType::channeltype > channels )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
channels	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

input	input signal pointer
output	output signal pointer
samples	number of samples
addition	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)
a	y coefficients of transfer function
b	x coefficients of transfer function
order	filter order (value of the highest exponent) without including delay term
a_delay	y delay coefficients of transfer function
b_delay	x delay coefficients of transfer function
delay	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()

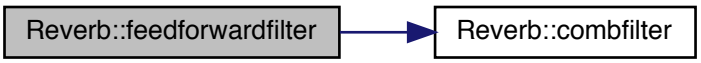
```
void Reverb::feedforwardfilter (
            float * input,
            float * output,
            int samples,
            bool addition,
            float gain,
            float original,
            int delay )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
addition	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)
gain	function transfer gain
original	gain of the original signal
delay	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,
```

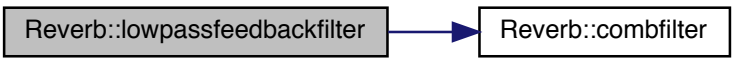
```
            bool addition,
            float gain,
            float rs,
            float d,
            int delay )
```

Parameters

input	input signal pointer
output	output signal pointer
samples	number of samples
addition	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)
gain	function transfer gain
rs	feed forward comb filter gain
d	low pass filter gain
delay	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

input	input signal pointer
output	output signal pointer
samples	number of samples
addition	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)
gain	function transfer gain
g	all pass filter gain
delay	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

input	input signal pointer
output	output signal pointer
samples	number of samples
addition	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)
gain	function transfer gain
g	all pass filter gain
delay	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

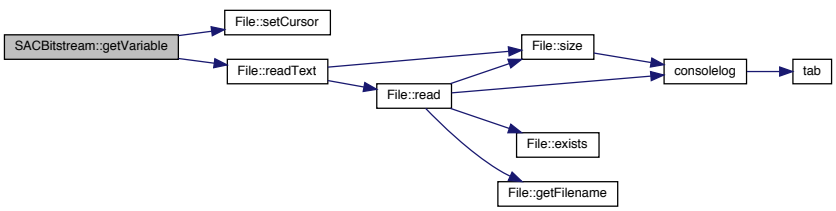
position	position in bits
length	number of bits

Returns

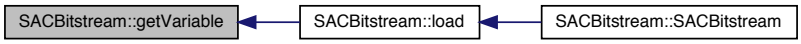
value of the bitstream variable

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:

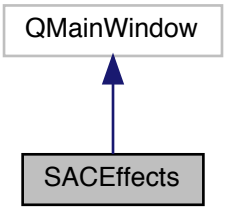
- src/sac/SACBitstream.h
- src/sac/SACBitstream.cpp

6.29 SACEffects Class Reference

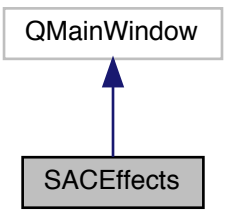
SACEffects window interface.

```
#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 enability 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.

6.29.2 Constructor & Destructor Documentation

6.29.2.1 SACEffects()

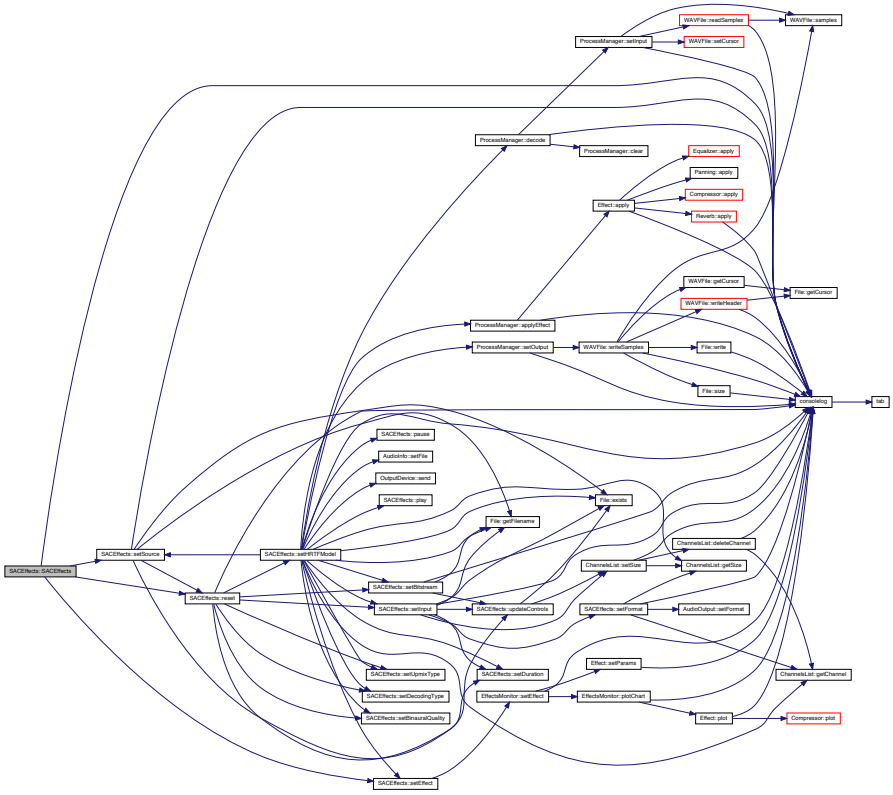
```
SACEffects::SACEffects (
            QWidget * framework = 0 )  [explicit]
```

Parameters

framework	SACEffects user interface object

Definition at line 12 of file SACEffects.cpp.

Here is the call graph for this function:



6.29.3 Member Function Documentation

6.29.3.1 setBinauralQuality()

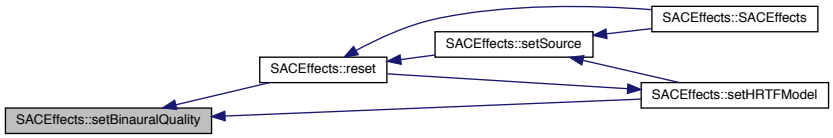
```
void SACEffects::setBinauralQuality (
            BinauralQuality::binauralquality binauralquality )
```

Parameters

binauralquality	binaural quality

Definition at line 401 of file SACEffects.cpp.

Here is the caller graph for this function:



6.29.3.2 setBitstream()

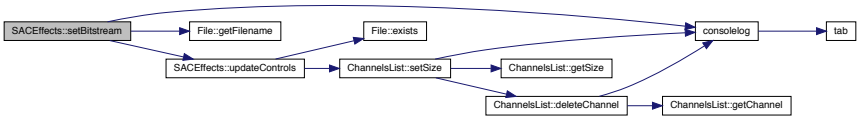
```
void SACEffects::setBitstream (
            std::string filename )
```

Parameters

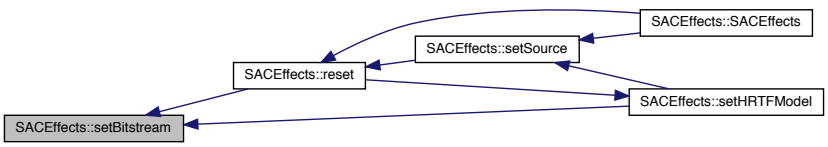
filename	file path

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()

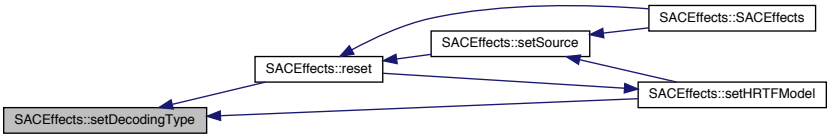
```
void SACEffects::setDecodingType (
            DecodingType::decodingtype decodingtype )
```

Parameters

decodingtype	decoding type

Definition at line 380 of file SACEffects.cpp.

Here is the caller graph for this function:



6.29.3.4 setDuration()

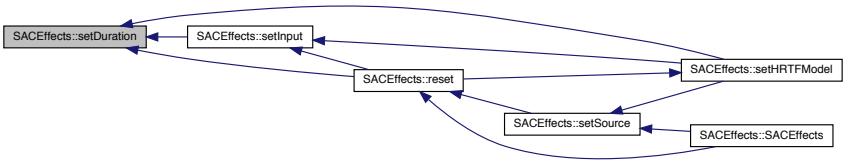
```
void SACEffects::setDuration (
            QLabel * label,
            double duration )
```

Parameters

label	user interface object where to indicate duration
duration	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()

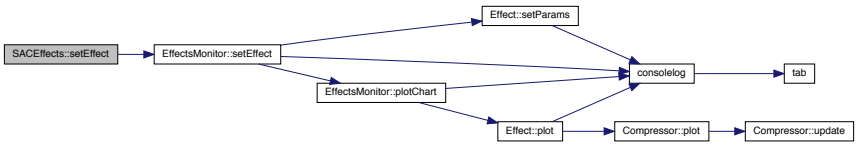
```
void SACEffects::setEffect (
            Effect::effectID effect )
```

Parameters

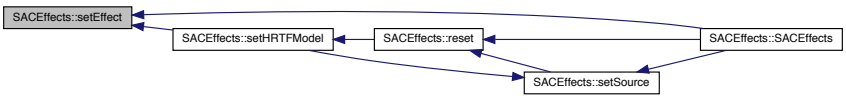
effect	selected effect

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()

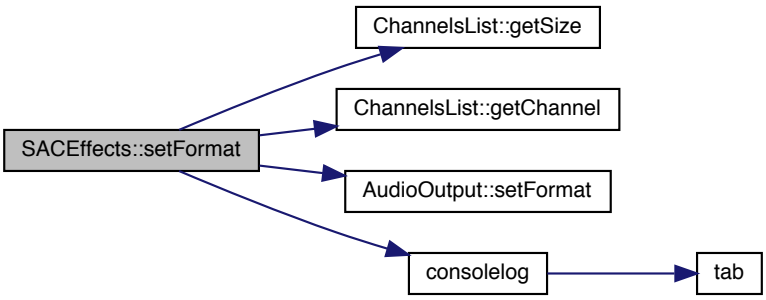
```
void SACEffects::setFormat (
            int fs,
            int samplesize )
```

Parameters

fs	signal sampling frequency
samplesize	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()

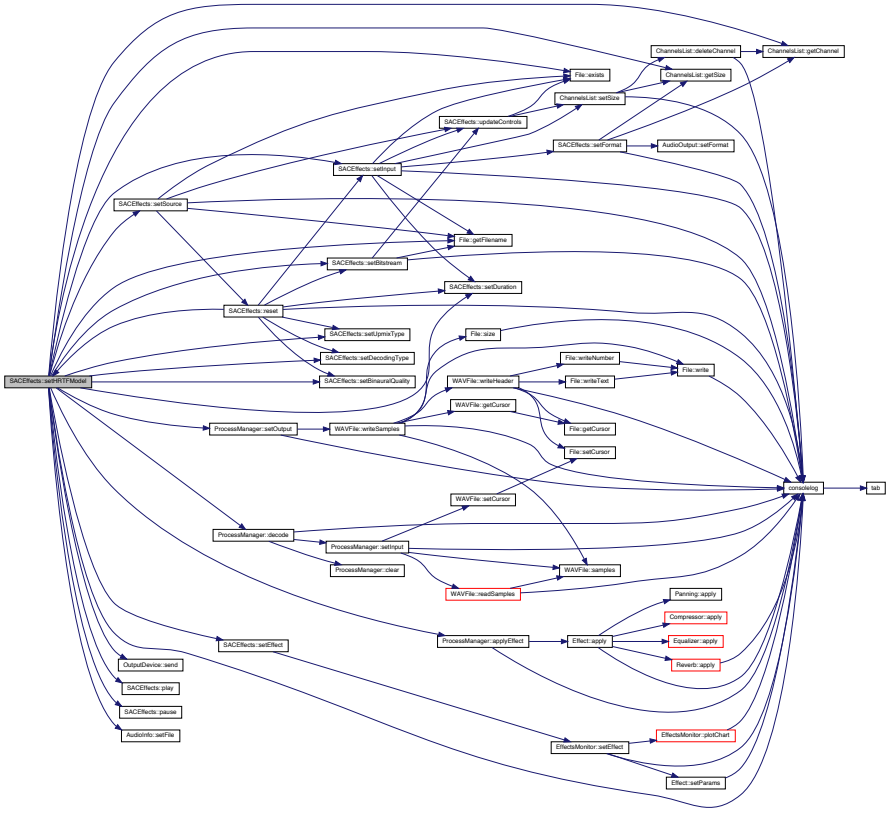
```
void SACEffects::setHRTFModel (
            HRTFModel::hrtfmodel hrtfmodel )
```

Parameters

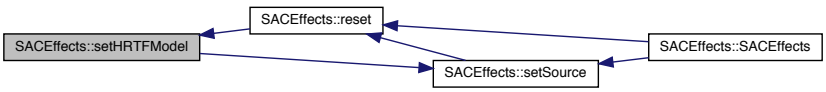
hrtfmodel	HRTF model

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()

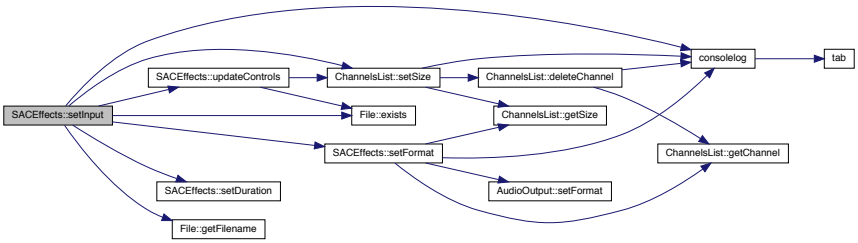
```
void SACEffects::setInput (
            std::string filename )
```

Parameters

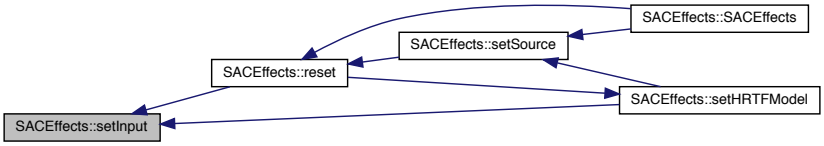
filename	file path

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()

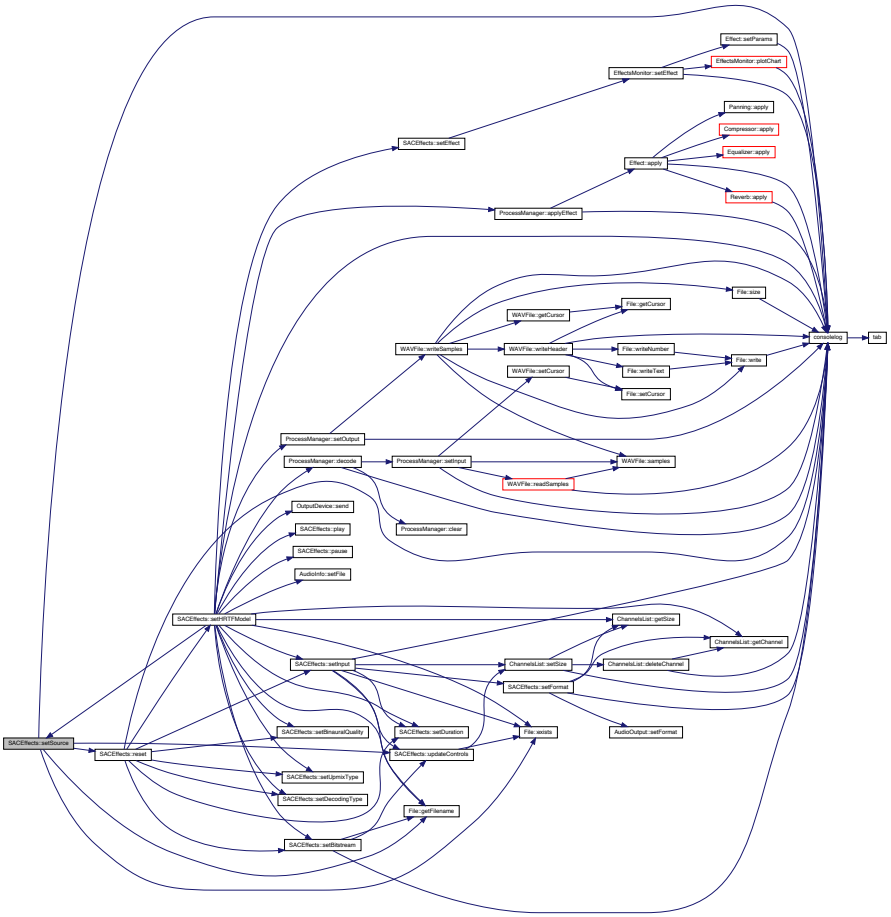
```
void SACEffects::setSource (
            std::string filename )
```

Parameters

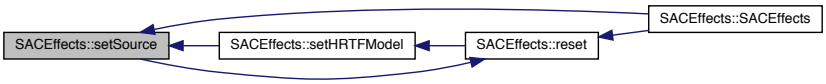
filename	file path

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()

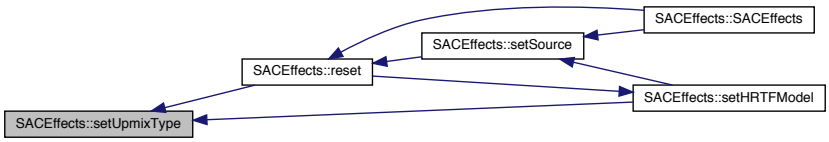
```
void SACEffects::setUpmixType (
            UpmixType::upmixtype upmixtype )
```

Parameters

upmixtype	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:

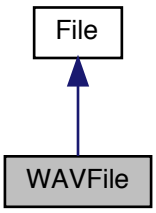
- src/interface/SACEffects.h

6.31 WAVFile Class Reference

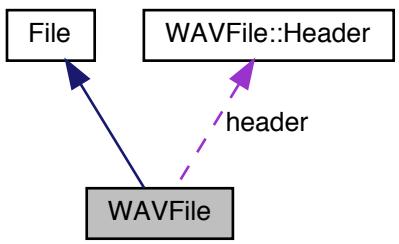
Audio file as WAV format class.

`#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

writepermission	file write permission (true if it is allowed)

Definition at line 220 of file File.cpp.

6.31.2.2 WAVFile() [2/3]

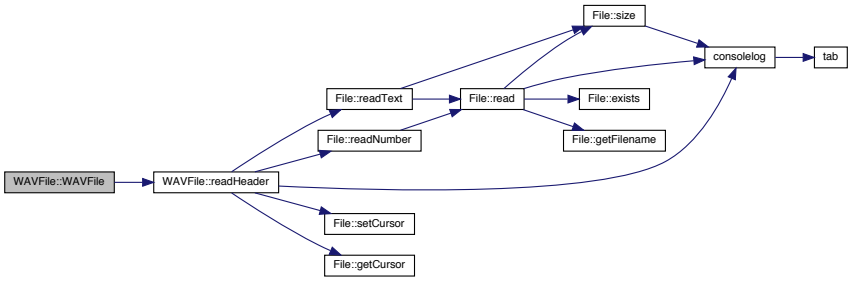
```
WAVFile::WAVFile (
            std::string filename,
            bool writepermission )
```

Parameters

filename	file path
writepermission	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]

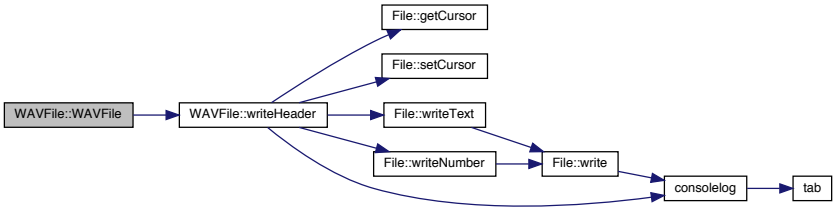
```
WAVFile::WAVFile (
            std::string filename,
            int channels,
            int fs,
            int sampleformat )
```

Parameters

filename	file path
channels	number of channels
fs	signal sample rate
sampleformat	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()

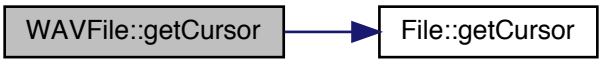
```
int WAVFile::getCursor ( )
```

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

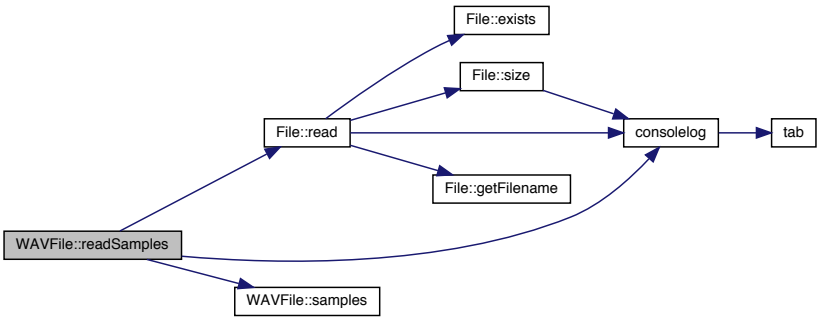
samples	number of samples

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()

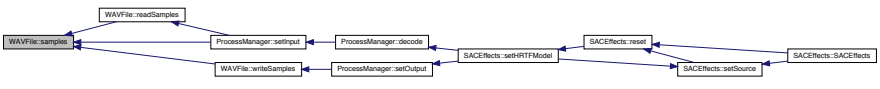
```
int WAVFile::samples ( )
```

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()

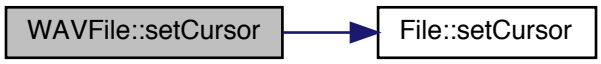
```
void WAVFile::setCursor (
            int cursor )
```

Parameters

cursor	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()

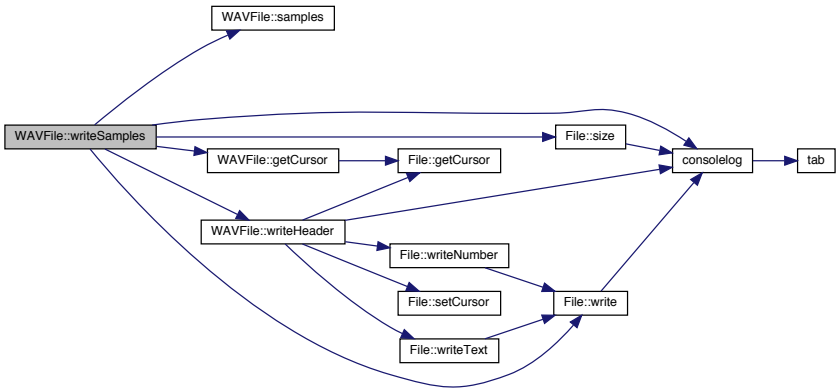
```
void WAVFile::writeSamples (
            float ** array,
            int samples )
```

Parameters

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

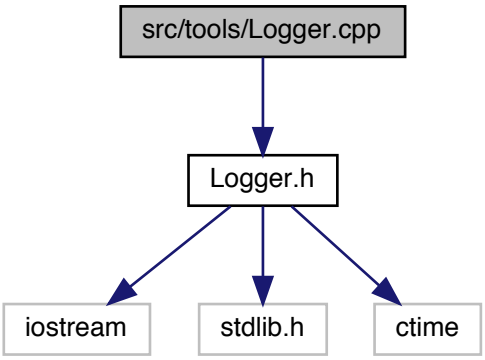
File Documentation

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

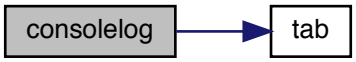
source	origin class/method/file where the message was logged
logtype	type of message
message	message

Returns

void

Definition at line 51 of file Logger.cpp.

Here is the call graph for this function:



7.1.2.2 tab()

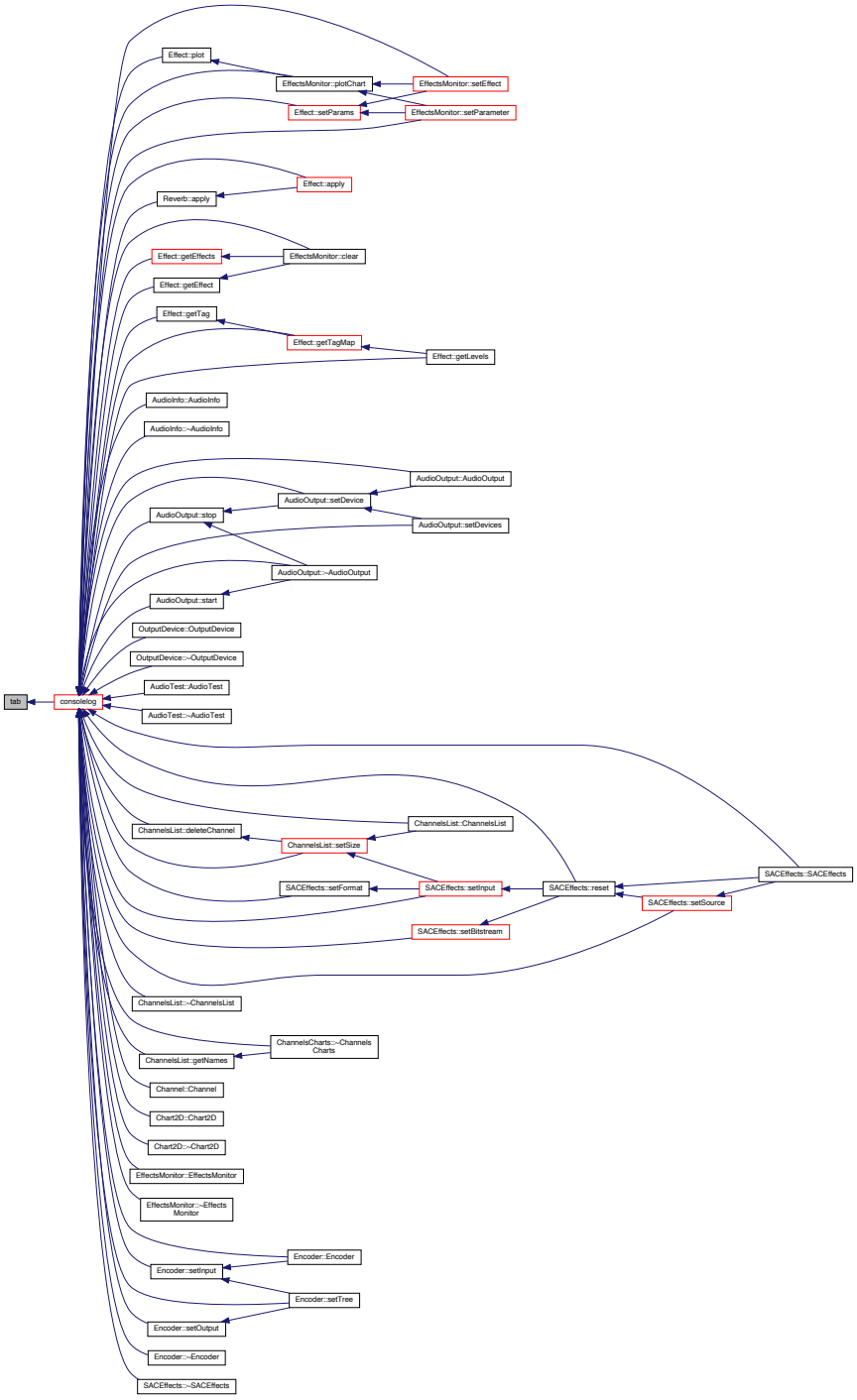
```
std::string tab (
            std::string content,
            const int tab_max )
```

Parameters

content	Content of the tabulation
tab_max	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.