SFML

* [Main Page](http://docs.google.com/index.htm)
* [Modules](http://docs.google.com/modules.htm)
* [Classes](http://docs.google.com/annotated.htm)
* [Files](http://docs.google.com/files.htm)
* [Class List](http://docs.google.com/annotated.htm)
* [Class Index](http://docs.google.com/classes.htm)
* [Class Hierarchy](http://docs.google.com/hierarchy.htm)
* [Class Members](http://docs.google.com/functions.htm)
* **sf**
* [SoundRecorder](http://docs.google.com/classsf_1_1SoundRecorder.htm)

[Public Member Functions](#_gjdgxs) | [Static Public Member Functions](#_30j0zll) | [Protected Member Functions](#_1fob9te) | [List of all members](http://docs.google.com/classsf_1_1SoundRecorder-members.htm)

sf::SoundRecorder Class Referenceabstract

[Audio module](http://docs.google.com/group__audio.htm)

Abstract base class for capturing sound data. [More...](http://docs.google.com/classsf_1_1SoundRecorder.htm#details)

#include <[SoundRecorder.hpp](http://docs.google.com/SoundRecorder_8hpp_source.htm)>

Inheritance diagram for sf::SoundRecorder:



| Public Member Functions | |
| --- | --- |
| virtual | [~SoundRecorder](http://docs.google.com/classsf_1_1SoundRecorder.htm#acc599e61aaa47edaae88cf43f0a43549) () |
|  | destructor |
|  | |
| void | [start](http://docs.google.com/classsf_1_1SoundRecorder.htm#a777e633114f7221cd7554a6ed486259e) (unsigned int sampleRate=44100) |
|  | Start the capture. |
|  | |
| void | [stop](http://docs.google.com/classsf_1_1SoundRecorder.htm#a8d9c8346aa9aa409cfed4a1101159c4c) () |
|  | Stop the capture. |
|  | |
| unsigned int | [getSampleRate](http://docs.google.com/classsf_1_1SoundRecorder.htm#a1f3726cbe0a2b2b291b36beea57960d7) () const |
|  | Get the sample rate. |
|  | |

| Static Public Member Functions | |
| --- | --- |
| static bool | [isAvailable](http://docs.google.com/classsf_1_1SoundRecorder.htm#aab2bd0fee9e48d6cfd449b1cb078ce5a) () |
|  | Check if the system supports audio capture. |
|  | |

| Protected Member Functions | |
| --- | --- |
|  | [SoundRecorder](http://docs.google.com/classsf_1_1SoundRecorder.htm#a50ebad413c4f157408a0fa49f23212a9) () |
|  | Default constructor. |
|  | |
| virtual bool | [onStart](http://docs.google.com/classsf_1_1SoundRecorder.htm#a7af418fb036201d3f85745bef78ce77f) () |
|  | Start capturing audio data. |
|  | |
| virtual bool | [onProcessSamples](http://docs.google.com/classsf_1_1SoundRecorder.htm#a2670124cbe7a87c7e46b4840807f4fd7) (const Int16 \*samples, std::size\_t sampleCount)=0 |
|  | Process a new chunk of recorded samples. |
|  | |
| virtual void | [onStop](http://docs.google.com/classsf_1_1SoundRecorder.htm#aefc36138ca1e96c658301280e4a31b64) () |
|  | Stop capturing audio data. |
|  | |

## Detailed Description

Abstract base class for capturing sound data.

[sf::SoundBuffer](http://docs.google.com/classsf_1_1SoundBuffer.htm) provides a simple interface to access the audio recording capabilities of the computer (the microphone).

As an abstract base class, it only cares about capturing sound samples, the task of making something useful with them is left to the derived class. Note that SFML provides a built-in specialization for saving the captured data to a sound buffer (see [sf::SoundBufferRecorder](http://docs.google.com/classsf_1_1SoundBufferRecorder.htm)).

A derived class has only one virtual function to override:

* onProcessSamples provides the new chunks of audio samples while the capture happens

Moreover, two additionnal virtual functions can be overriden as well if necessary:

* onStart is called before the capture happens, to perform custom initializations
* onStop is called after the capture ends, to perform custom cleanup

The audio capture feature may not be supported or activated on every platform, thus it is recommended to check its availability with the [isAvailable()](http://docs.google.com/classsf_1_1SoundRecorder.htm#aab2bd0fee9e48d6cfd449b1cb078ce5a) function. If it returns false, then any attempt to use an audio recorder will fail.

It is important to note that the audio capture happens in a separate thread, so that it doesn't block the rest of the program. In particular, the onProcessSamples and onStop virtual functions (but not onStart) will be called from this separate thread. It is important to keep this in mind, because you may have to take care of synchronization issues if you share data between threads.

Usage example:

class CustomRecorder : public [sf::SoundRecorder](http://docs.google.com/classsf_1_1SoundRecorder.htm)

{

virtual bool [onStart](http://docs.google.com/classsf_1_1SoundRecorder.htm#a7af418fb036201d3f85745bef78ce77f)() // optional

{

// Initialize whatever has to be done before the capture starts

...

// Return true to start playing

return true;

}

virtual bool [onProcessSamples](http://docs.google.com/classsf_1_1SoundRecorder.htm#a2670124cbe7a87c7e46b4840807f4fd7)(const Int16\* samples, std::size\_t sampleCount)

{

// Do something with the new chunk of samples (store them, send them, ...)

...

// Return true to continue playing

return true;

}

virtual void [onStop](http://docs.google.com/classsf_1_1SoundRecorder.htm#aefc36138ca1e96c658301280e4a31b64)() // optional

{

// Clean up whatever has to be done after the capture ends

...

}

}

// Usage

if (CustomRecorder::isAvailable())

{

CustomRecorder recorder;

recorder.start();

...

recorder.stop();

}

See Also[sf::SoundBufferRecorder](http://docs.google.com/classsf_1_1SoundBufferRecorder.htm)

Definition at line [42](http://docs.google.com/SoundRecorder_8hpp_source.htm#l00042) of file [SoundRecorder.hpp](http://docs.google.com/SoundRecorder_8hpp_source.htm).

## Constructor & Destructor Documentation

| | virtual sf::SoundRecorder::~SoundRecorder | ( |  | ) |  | | --- | --- | --- | --- | --- | | virtual |
| --- | --- | --- | --- | --- | --- | --- |

destructor

| | sf::SoundRecorder::SoundRecorder | ( |  | ) |  | | --- | --- | --- | --- | --- | | protected |
| --- | --- | --- | --- | --- | --- | --- |

Default constructor.

This constructor is only meant to be called by derived classes.

## Member Function Documentation

| unsigned int sf::SoundRecorder::getSampleRate | ( |  | ) | const |
| --- | --- | --- | --- | --- |

Get the sample rate.

The sample rate defines the number of audio samples captured per second. The higher, the better the quality (for example, 44100 samples/sec is CD quality).

ReturnsSample rate, in samples per second

| | static bool sf::SoundRecorder::isAvailable | ( |  | ) |  | | --- | --- | --- | --- | --- | | static |
| --- | --- | --- | --- | --- | --- | --- |

Check if the system supports audio capture.

This function should always be called before using the audio capture features. If it returns false, then any attempt to use [sf::SoundRecorder](http://docs.google.com/classsf_1_1SoundRecorder.htm) or one of its derived classes will fail.

ReturnsTrue if audio capture is supported, false otherwise

| | virtual bool sf::SoundRecorder::onProcessSamples | ( | const Int16 \* | *samples*, | | --- | --- | --- | --- | |  |  | std::size\_t | *sampleCount* | |  | ) |  |  | | protectedpure virtual |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |

Process a new chunk of recorded samples.

This virtual function is called every time a new chunk of recorded data is available. The derived class can then do whatever it wants with it (storing it, playing it, sending it over the network, etc.).

Parameters

| samples | Pointer to the new chunk of recorded samples |
| --- | --- |
| sampleCount | Number of samples pointed by *samples* |

ReturnsTrue to continue the capture, or false to stop it

Implemented in [sf::SoundBufferRecorder](http://docs.google.com/classsf_1_1SoundBufferRecorder.htm#a9ceb94de14632ae8c1b78faf603b4767).

| | virtual bool sf::SoundRecorder::onStart | ( |  | ) |  | | --- | --- | --- | --- | --- | | protectedvirtual |
| --- | --- | --- | --- | --- | --- | --- |

Start capturing audio data.

This virtual function may be overriden by a derived class if something has to be done every time a new capture starts. If not, this function can be ignored; the default implementation does nothing.

ReturnsTrue to start the capture, or false to abort it

Reimplemented in [sf::SoundBufferRecorder](http://docs.google.com/classsf_1_1SoundBufferRecorder.htm#a531a7445fc8a48eaf9fc039c83f17c6f).

| | virtual void sf::SoundRecorder::onStop | ( |  | ) |  | | --- | --- | --- | --- | --- | | protectedvirtual |
| --- | --- | --- | --- | --- | --- | --- |

Stop capturing audio data.

This virtual function may be overriden by a derived class if something has to be done every time the capture ends. If not, this function can be ignored; the default implementation does nothing.

Reimplemented in [sf::SoundBufferRecorder](http://docs.google.com/classsf_1_1SoundBufferRecorder.htm#ab8e53849312413431873a5869d509f1e).

| void sf::SoundRecorder::start | ( | unsigned int | *sampleRate* = 44100 | ) |  |
| --- | --- | --- | --- | --- | --- |

Start the capture.

The *sampleRate* parameter defines the number of audio samples captured per second. The higher, the better the quality (for example, 44100 samples/sec is CD quality). This function uses its own thread so that it doesn't block the rest of the program while the capture runs. Please note that only one capture can happen at the same time.

Parameters

| sampleRate | Desired capture rate, in number of samples per second |
| --- | --- |

See Also[stop](http://docs.google.com/classsf_1_1SoundRecorder.htm#a8d9c8346aa9aa409cfed4a1101159c4c)

| void sf::SoundRecorder::stop | ( |  | ) |  |
| --- | --- | --- | --- | --- |

Stop the capture.

See Also[start](http://docs.google.com/classsf_1_1SoundRecorder.htm#a777e633114f7221cd7554a6ed486259e)

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

* [SoundRecorder.hpp](http://docs.google.com/SoundRecorder_8hpp_source.htm)

Copyright � Laurent Gomila  ::  Documentation generated by [doxygen](http://www.doxygen.org/)  ::