



## z.reverb SDK Documentation

Alexander Lerch

(c) 2006 by zplane.development

July 11, 2006

## Contents

<b>1 z.reverb SDK Documentation</b>	<b>2</b>
1.1 Introduction	2
1.2 API Documentation	2
1.2.1 Naming Conventions	2
1.2.2 Required Functions	2
1.2.3 Optional Functions	3
1.2.4 Parameter Setting Functions	3
1.2.5 Calling Conventions	8
1.3 MSVC6 Workspace Reverb.dsw	9
1.3.1 File Structure	9
1.3.2 Project Structure	10
1.3.3 Project Configurations	10
1.4 Coding Style minimal overview	10
1.5 Graph Element descriptions	10
1.6 Support	11
<b>2 z.reverb SDK Directory Hierarchy</b>	<b>12</b>
2.1 z.reverb SDK Directories	12
<b>3 z.reverb SDK File Index</b>	<b>12</b>
3.1 z.reverb SDK File List	12
<b>4 z.reverb SDK Directory Documentation</b>	<b>12</b>
4.1 incl/ Directory Reference	12
<b>5 z.reverb SDK File Documentation</b>	<b>13</b>
5.1 docugen.txt File Reference	13
5.1.1 Detailed Description	13
5.2 ReverbAPI.h File Reference	13
5.2.1 Detailed Description	13
5.2.2 Define Documentation	14
5.2.3 Typedef Documentation	15
5.2.4 Enumeration Type Documentation	15
5.2.5 Function Documentation	16

# 1 z.reverb SDK Documentation

## 1.1 Introduction

The z.reverb SDK allows to add a reverberation effect to the incoming audio material.

The first section of this document gives a detailed API documentation with input and output requirements and a detailed function documentation. After that, a short overview over the z.reverb-SDK folder structure is given.

The following chapters contain automatically generated function and parameter description.

## 1.2 API Documentation

To integrate the reverb in source code, the header file for this effect has to be included. The name of the header file is **ReverbAPI.h**. Furthermore, the library libReverb.lib has to be linked to the executable or library.

The interface allows in-place processing, i.e. the audio data can be replaced by the audio data plus the corresponding effect. Note that inplace processing is only possible if the number of input channels equals the number of output channels.

All needed variable types are either standard C-types or are defined in the API-header files.

In the case of more than one input channel, the input data has to be *interleaved*.

### 1.2.1 Naming Conventions

All API functions are preceded by an initial *Reverb\_* to be easily distinguished from other functions. After that, the function name makes clear what should happen; all functions beginning with *Reverb\_Set* do set any properties or parameters of the corresponding effect. All functions ending with *Instance* create or destroy an object/instance of the effect. Examples are *Reverb\_CreateInstance*, *Reverb\_GetReverbTime*, *Reverb\_Process*.

When talking about **frames**, the number of audio samples in one channel is meant. If the sample size is float, one sample has a memory usage of 4 byte. The memory usage of one stereo frame is 8 byte.

### 1.2.2 Required Functions

The following functions have to be called when using an effect of the z.reverb library:

- **Reverb\_CreateInstance** (**void\*\* phInstance**, **int iSampleRate**, **int iNumOfChannels**, **Quality\_t qQualityMode**)

This function creates an instance of the required effects. The parameter *phInstance* is the address of a pointer where the handle to the instance is written, *iSampleRate* is sample rate of input signal in Hz, *iNumOfChannels* is the number

of input channels, and `qQualityMode` defines four different quality levels, as defined in `Quality_t`. The lowest quality level will give the lowest processing workload and vice versa.

- **Reverb\_DestroyInstance** (`void** phInstance`)

This function destroys an instance of the used effect. The parameter `phInstance` is the address of the instance handle pointer. The instance handle will be set to `NULL`.

The return value will equal 0 on success.

- **Reverb\_Process** (`void* pCReverbInstance`, `float* pInputBuffer`, `float* pOutputBuffer`, `int iNumOfFrames`)

This function destroys an instance of the used effects. Available functions are: The parameter `phInstance` is the instance handle pointer, `pfInputBuffer` is the pointer to interleaved input data, `pfOutputBuffer` is pointer to buffer for the interleaved output data (may be the same as input data in all cases except the case that the number of output channels does not equal the number of input channels), `iNumOfFrames` is the number of frames in the input buffer. Note that the number of output channels is always two!

The return value will equal 0 on success.

### 1.2.3 Optional Functions

Although the use of the functions listed here is optional, many of these are either required to use this library in a sensible way or are recommended to use.

- **Reverb\_Reset** ()

This function resets the internal buffers of the used effects. It should be called when the same effect instance should process a different input audio stream, or when the the effect was bypassed. The parameter `phInstance` is the instance handle pointer.

The return value will equal 0 on success.

### 1.2.4 Parameter Setting Functions

There are plenty of parameters to control the reverberation effect. However, it does not make sense in every use-case to provide all these parameters in the user interface. From a developers point of view, the parameters could be clustered in different priority classes, i.e. parameters which have to appear in user interface, and parameters with lower priority that do not necessarily have to appear in the user interface. This following sorts the available parameters into three priority classes:

<i>High Priority:</i>	Preset, ReverbTime, PreDelay, Wetness (only when used as insert), HighCut
<i>Medium Priority:</i>	LPAmount, ERGain, LRGain, BassBoost
<i>Low Priority:</i>	Liveness, Diffusion, Wetness (when used as send)

The API provides the following parameter functions:

- **Reverb\_GetBassBoost** (.)

Returns the bass gain. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the gain of the bass frequencies in dB. See also function [Reverb\\_SetBassBoost](#) (.).

- **Reverb\_GetDiffusion** (.)

Returns the diffusivity. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the diffusivity in percent. See also function [Reverb\\_SetDiffusion](#) (.).

- **Reverb\_GetERGain** (.)

Returns the gain of the early reflections. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the gain of the early reflections in dB. See also function [Reverb\\_SetERGain](#) (.).

- **Reverb\_GetHold** (.)

Returns the status of the hold effect. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the current status of HOLD. See also function [Reverb\\_SetHold](#) (.).

- **Reverb\_GetLiveness** (.)

Returns the liveliness. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the liveliness in percent. See also function [Reverb\\_SetLiveness](#) (.).

- **Reverb\_GetLRGain** (.)

Returns the gain of the late reverberation tail. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the gain of the late reverberation tail in dB. See also function [Reverb\\_SetLRGain](#) (.).

- **Reverb\_GetPreDelay** (.)

Returns the pre delay. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

The return value is the pre delay in seconds. See also function [Reverb\\_SetPreDelay](#) (.).

- [Reverb\\_GetPreset](#) (.)

Returns the current preset. The function parameters are:

- **void\*** **pCReverbInstance**: instance handle pointer.

The return value is the current preset. See also function [Reverb\\_SetPreset](#) (.).

- [Reverb\\_GetReverbTime](#) (.)

Returns the reverberation time. The function parameters are:

- **void\*** **pCReverbInstance**: instance handle pointer.

The return value is the reverberation time in seconds. See also function [Reverb\\_SetReverbTime](#) (.).

- [Reverb\\_GetWetness](#) (.)

Returns the wetness. The function parameters are:

- **void\*** **pCReverbInstance**: instance handle pointer.

The return value is the wetness in percent. See also function [Reverb\\_SetWetness](#) (.).

- [Reverb\\_SetBassBoost](#) (.)

Sets the gain for the bass frequencies. The function parameters are:

- **void\*** **pCReverbInstance**: instance handle pointer.
- **float fBassBoost**: gain of the bass frequencies in the reverberation in dB. The allowed range is from -18...18, default value depends on the selected preset. High values will result in a darker room (and in an increased output gain).

The return value will equal 0 on success.

- [Reverb\\_SetDiffusion](#) (.)

Sets the diffusivity of the reverberation. The function parameters are:

- **void\*** **pCReverbInstance**: instance handle pointer.
- **float fDiffusion**: diffusivity of the the reverberation in percent. The allowed range is from 0...1, default value depends on the selected preset. High values will result in a larger room impression.

The return value will equal 0 on success.

- [Reverb\\_SetERGain](#) (.)

Sets the gain for early reflections. The function parameters are:

- **void\*** **pCReverbInstance**: instance handle pointer.

- **float fERGainIndB**: gain of the early reflections in the reverberation in dB. The allowed range is from -12...12, default value depends on the selected preset. High values will result in a more narrowed room impression (and in an increased output gain).

The return value will equal 0 on success.

```
int Reverb_SetERLRBalance (void* , float );
```

- **Reverb\_SetERLRBalance (.)**

Sets the amount of ER/LR gain relative to the current ERGain/LRGain-Settings. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fERLRBalance**: value between -1 (maximum ER, minimum LR) and 1 (minimum ER, maximum LR), default value is 0 (changes depend on current ERGain/LRGain settings!) on the selected preset. High values will result in a more narrowed room impression (and in an increased output gain).

The return value will equal 0 on success.

- **Reverb\_SetHighCut (.)**

Sets the low pass frequency for the reverberation. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fLPFreq**: low pass frequency of the reverberation in percent of the sample rate. The allowed range is from 0.0005...0.9995, default value depends on the selected preset. Low values will result in a darker room (and in a decreased output gain). See also function [Reverb\\_SetLPAmount \(.\)](#)

The return value will equal 0 on success.

- **Reverb\_SetHold (.)**

This function is not motivated by room acoustics, but is an effect function that allows to freeze the reverberation at one state. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **Hold\_t eHold**: hold mode can be Hold\_tON to switch the effect on, Hold\_tOFF to switch the effect off or Hold\_tACCUMULATE to switch the effect on but also send new input to the reverb. The last option is experimental and will result in clipping, i.e. in instability in most cases, especially in the case of loud input signals.

The return value will equal 0 on success.

- **Reverb\_SetLiveness (.)**

Sets the liveness of of the reverberation tails, i.e. could make the reverberation more interesting. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.

- **float fLiveness**: amount of liveness of the reverberation in percent. The allowed range is from 0.05...1.0, default value depends on the selected preset. High values could result in detuning effects especially for tonal input signals.

The return value will equal 0 on success.

- **Reverb\_SetLPAmount** (.)

Sets the amount of low pass filtering for the reverberation. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fLPAmount**: amount of low pass filtering of the reverberation. The allowed range is from 0.0005...0.9995, default value depends on the selected preset. High values will result in a darker room (and in a decreased output gain). See also function **Reverb\_SetHighCut** (.)

The return value will equal 0 on success.

- **Reverb\_SetLRGain** (.)

Sets the gain for late reverberation. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fLRGainIndB**: gain of the early reflections in the reverberation in dB. The allowed range is from -12...12, default value depends on the selected preset. High values will result in a more open room impression (and in an increased output gain).

The return value will equal 0 on success.

- **Reverb\_SetPreDelay** (.)

Sets the predelay, i.e. the delay wet output signal, of the effect. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fPreDelay**: delay of the wet output signal in seconds. The allowed range is from 0...0.3, default value depends on the selected preset.

The return value will equal 0 on success.

- **Reverb\_SetPreset** (.)

Sets all intern parameters acc. to the preset. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **ReverbPresets\_t ePreset**: preset to set acc. the enum **ReverbPresets\_t**. The parameter ePreset can be of the type **\_MEDIUM\_HALL**, **\_LARGE\_HALL**, **\_PLATE**, **\_CATHEDRAL** or **\_ROOM**. Default value is **\_MEDIUM\_HALL**.

The return value will equal 0 on success.

- **Reverb\_SetReverbTime** (.)

Sets the reverberation time (the length of the reverberation tail). The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fReverbTime**: reverberation time in seconds. The allowed range is from 0.4986...1000, default value depends on the selected preset. High values will result in a larger room impression.

The return value will equal 0 on success.

- **Reverb\_SetWetness** (.)

Sets the wetness of the effect. The function parameters are:

- **void\* pCReverbInstance**: instance handle pointer.
- **float fWetness**: amount of the wet signal in percent of the output signal. The allowed range is from 0...1.0, default value is 0.2. The default value corresponds to a dB value of -12dB relative to the direct signal (ratio 0.2/0.8).

The return value will equal 0 on success.

### 1.2.5 Calling Conventions

The usage of the effects is straight forward. After an instance has been created with **Reverb\_CreateInstance**, the parameters can be set with the functions *EffectName\_SetParameterName* (.).

Then the audio signal can be processed with **Reverb\_Process**. Please note that the parameter settings functions must not be called while the processing function is called. However, they may be called between two calls of the processing function.

After successful processing, the instance of the effect can be destroyed with **Reverb\_DestroyInstance**. The following example can be found in the file TestReverb.cpp.

Declare a new handle of a reverb instance:

```
void                *pReverb;                // instance handle
```

and create a new instance with the necessary information

```
//create a new instance of reverberation
Reverb_CreateInstance(&pReverb, sfInputInfo.samplerate, sfInputInfo.channels, _HIGH );
```

After that, you are able to set and get parameters, e.g.

```
Reverb_SetPreset (pReverb, _CATHEDRAL);
fprintf(stderr, "Reverb Time(s):\t%2.2f\n", Reverb_GetReverbTime (pReverb));
fprintf(stderr, "EarlyRefl(dB):\t%2.2f\n", Reverb_GetERGain (pReverb));
fprintf(stderr, "LateReverb(dB):\t%2.2f\n", Reverb_GetLRGain (pReverb));
fprintf(stderr, "PreDelay(ms):\t%2.2f\n", 1000.0F*Reverb_GetPreDelay (pReverb));
```

The actual processing is done with succeeding blocks of audio data:

```
// now we can begin to process!
while (bReadNextFrame)
{
    iNumSamplesRead = (int)(sf_read_float (pFInputFile, afInputData, (iBlocksizeFile)));
    iNumSamplesRead /= sfInputInfo.channels;

    Reverb_Process (pReverb, afInputData, afOutputData, iNumSamplesRead);
} // end of process loop
```

After successful processing, the instance can be destroyed:

```
Reverb_DestroyInstance (&pReverb);
```

## 1.3 MSVC6 Workspace Reverb.dsw

### 1.3.1 File Structure

**1.3.1.1 Documentation** This documentation and all other documentation can be found in the directory **./doc**.

**1.3.1.2 Project Files** The MS VisualC++-Workspace (.dsw) and all Projectfiles (.dsp) can be found in the directory **./build** and its subfolders, where the subfolders names correspond to the project names.

**1.3.1.3 Source Files** All source files are in the directory **./src** and its subfolders, where the subfolder names equally correspond to the project names.

**1.3.1.4 Include Files** If include files are project intern, they are in the source directory **./src** of the project itself. If include files are to be included by other projects they can be found in **./src/include**. If a SDK-like library/DLL is built for which a header is needed, such a header can be found in **./inc**.

**1.3.1.5 Resource Files** The resource files can be found in the subdirectory **/res** of the corresponding build-directory.

**1.3.1.6 Library Files** The directory **./lib** is for used and built libraries.

**1.3.1.7 Binary Files** The final executable as well as the distributable Dynamic Link Libraries can be found in the directory **./bin**. In debug-builds, the binary files are in the subfolder **/Debug**.

**1.3.1.8 Temporary Files** The directory **./tmp** is for all temporary files while building the projects. In debug-builds, the temporary files can be found in the subfolder **/Debug**.

### 1.3.2 Project Structure

The project structure is as following:

- **TestReverb**: command line test project for the reverb.

The project output is an executable binary (EXE).

The open source libraries libSndFile.lib as audio file IO library and libPortAudio.lib for real-time audio output could be used. These additional libraries are not required for the libReverb library and must only be used compliant to their respective licenses.

### 1.3.3 Project Configurations

For all projects included in the workspace, the default configurations Win32 Release and Win32 Debug are available.

## 1.4 Coding Style minimal overview

Variable names have a preceding letter indicating their types:

unsigned:	u
pointer:	p
array:	a
class:	C
bool:	b
char:	c
short (int16):	s
int (int32):	i
__int64:	l
float (float32):	f
double (float64):	d
char:	c

For example, a pointer to a buffer of unsigned ints will be named puiBufferName.

## 1.5 Graph Element descriptions

This is an excerpt from the doxygen documentation:

*The elements in the class diagrams in HTML and RTF have the following meaning:*

- *A yellow box indicates a class. A box can have a little marker in the lower right corner to indicate that the class contains base classes that are hidden. For the class diagrams the maximum tree width is currently 8 elements. If a tree is wider some nodes will be hidden. If the box is filled with a dashed pattern the inheritance relation is virtual.*

- A white box indicates that the documentation of the class is currently shown.
- A grey box indicates an undocumented class.
- A solid dark blue arrow indicates public inheritance.
- A dashed dark green arrow indicates protected inheritance.
- A dotted dark green arrow indicates private inheritance.

The elements in the class diagram in PDF have the following meaning:

- A white box indicates a class. A marker in the lower right corner of the box indicates that the class has base classes that are hidden. If the box has a dashed border this indicates virtual inheritance.
- A solid arrow indicates public inheritance.
- A dashed arrow indicates protected inheritance.
- A dotted arrow indicates private inheritance.

The elements in the graphs (...) have the following meaning:

- A white box indicates a class or struct or file.
- A box with a red border indicates a node that has more arrows than are shown! In other words: the graph is truncated with respect to this node. The reason why a graph is sometimes truncated is to prevent images from becoming too large. For the graphs generated with dot doxygen tries to limit the width of the resulting image to 1024 pixels.
- A black box indicates that the class' documentation is currently shown.
- A dark blue arrow indicates an include relation (for the include dependency graph) or public inheritance (for the other graphs).
- A dark green arrow indicates protected inheritance.
- A dark red arrow indicates private inheritance.
- A purple dashed arrow indicated a "usage" relation, the edge of the arrow is labeled with the variable(s) responsible for the relation. Class A uses class B, if class A has a member variable m of type C, where B is a subtype of C (e.g. C could be B, B\*, T<B>\* ).

## 1.6 Support

Support for the SDK is - within the limits of the agreement - available from:

[zplane.development](http://zplane.development)

katzbachstr. 21

d-10965 berlin  
germany  
fon: +49.30.854 09 15.0  
fax: +49.30.854 09 15.5  
@: [info@zplane.de](mailto:info@zplane.de)

## 2 z.reverb SDK Directory Hierarchy

### 2.1 z.reverb SDK Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

**incl** **12**

## 3 z.reverb SDK File Index

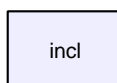
### 3.1 z.reverb SDK File List

Here is a list of all files with brief descriptions:

**ReverbAPI.h** (Interface of the zReverb effect) **13**

## 4 z.reverb SDK Directory Documentation

### 4.1 incl/ Directory Reference



#### Files

- file **ReverbAPI.h**  
*interface of the zReverb effect.*

## 5 z.reverb SDK File Documentation

### 5.1 docugen.txt File Reference

#### 5.1.1 Detailed Description

source documentation main file

Definition in file [docugen.txt](#).

### 5.2 ReverbAPI.h File Reference

#### 5.2.1 Detailed Description

interface of the zReverb effect.

:

Definition in file [ReverbAPI.h](#).

#### Defines

- #define [\\_\\_REVERBAPI\\_HEADER\\_INCLUDED\\_\\_](#)

#### Typedefs

- typedef enum [ReverbPresets\\_t\\_tag](#) [ReverbPresets\\_t](#)  
*List of available presets (names explain the preset).*
- typedef enum [Hold\\_t\\_tag](#) [Hold\\_t](#)  
*possible states of hold-effect*
- typedef enum [Quality\\_t\\_tag](#) [Quality\\_t](#)  
*possible quality modes to adjust workload vs. quality*

#### Enumerations

- enum [ReverbPresets\\_t\\_tag](#) {  
[\\_MEDIUM\\_HALL](#), [\\_LARGE\\_HALL](#), [\\_PLATE](#), [\\_CATHEDRAL](#),  
[\\_ROOM](#), [\\_ROOM1](#), [\\_ROOM2](#) }  
*List of available presets (names explain the preset).*
- enum [Hold\\_t\\_tag](#) { [\\_HOLD\\_ON](#), [\\_HOLD\\_OFF](#), [\\_HOLD\\_ACCUMULATE](#) }  
*possible states of hold-effect*

- enum `Quality_t_tag` { `_HIGHEST`, `_HIGH`, `_LOW`, `_LOWEST` }  
*possible quality modes to adjust workload vs. quality*

## Functions

- int `Reverb_CreateInstance` (void \*\*pCReverb, int iSampleRate, int iNumOfChannels, `Quality_t` qQualityMode)
- int `Reverb_DestroyInstance` (void \*\*pCReverb)
- void `Reverb_Reset` (void \*pCReverb)
- int `Reverb_Process` (void \*pCReverb, float \*pInputBuffer, float \*pOutputBuffer, int iNumOfFrames)
- int `Reverb_SetPreset` (void \*pCReverb, `ReverbPresets_t` ePreset)
- int `Reverb_SetWetness` (void \*pCReverb, float fWetness)
- int `Reverb_SetPreDelay` (void \*pCReverb, float fPreDelay)
- int `Reverb_SetBassBoost` (void \*pCReverb, float fBassBoost)
- int `Reverb_SetHighCut` (void \*pCReverb, float fLPFreq)
- int `Reverb_SetLPAmount` (void \*pCReverb, float fLPAmount)
- int `Reverb_SetReverbTime` (void \*pCReverb, float fReverbTime)
- int `Reverb_SetERLRBalance` (void \*pCReverb, float fERLRBalance)
- int `Reverb_SetERGain` (void \*pCReverb, float fERGainIndB)
- int `Reverb_SetLRGain` (void \*pCReverb, float fLRGainIndB)
- int `Reverb_SetHold` (void \*pCReverb, `Hold_t` eHold)
- int `Reverb_SetLiveness` (void \*pCReverb, float fLiveness)
- int `Reverb_SetDiffusion` (void \*pCReverb, float fDiffusion)
- float `Reverb_GetReverbTime` (void \*pCReverb)
- `Hold_t` `Reverb_GetHold` (void \*pCReverb)
- float `Reverb_GetERLRBalance` (void \*pCReverb)
- float `Reverb_GetERGain` (void \*pCReverb)
- float `Reverb_GetLRGain` (void \*pCReverb)
- float `Reverb_GetPreDelay` (void \*pCReverb)
- float `Reverb_GetBassBoost` (void \*pCReverb)
- float `Reverb_GetWetness` (void \*pCReverb)
- float `Reverb_GetLiveness` (void \*pCReverb)
- float `Reverb_GetDiffusion` (void \*pCReverb)
- float `Reverb_GetHighCut` (void \*pCReverb)
- float `Reverb_GetLPAmount` (void \*pCReverb)
- `ReverbPresets_t` `Reverb_GetPreset` (void \*pCReverb)

## 5.2.2 Define Documentation

### 5.2.2.1 #define \_\_REVERBAPI\_HEADER\_INCLUDED\_\_

Definition at line 116 of file ReverbAPI.h.

### 5.2.3 Typedef Documentation

#### 5.2.3.1 typedef enum **Hold\_t\_tag** **Hold\_t**

possible states of hold-effect

#### 5.2.3.2 typedef enum **Quality\_t\_tag** **Quality\_t**

possible quality modes to adjust workload vs. quality

#### 5.2.3.3 typedef enum **ReverbPresets\_t\_tag** **ReverbPresets\_t**

List of available presets (names explain the preset).

### 5.2.4 Enumeration Type Documentation

#### 5.2.4.1 enum **Hold\_t\_tag**

possible states of hold-effect

##### Enumerator:

***\_HOLD\_ON*** repeat the current reverberation state for infinity

***\_HOLD\_OFF*** stop repeating the current reverberation state for infinity

***\_HOLD\_ACCUMULATE*** as ***\_HOLD\_ON***, but add new samples to the reverb.

**\*\*ATTENTION\*\***: this will lead to clipping pretty soon!

Definition at line 133 of file ReverbAPI.h.

#### 5.2.4.2 enum **Quality\_t\_tag**

possible quality modes to adjust workload vs. quality

##### Enumerator:

***\_HIGHEST*** highest quality mode (enhanced stereo processing, default)

***\_HIGH*** high quality mode (previous default v1.x)

***\_LOW*** low quality mode (enhanced performance)

***\_LOWEST*** lowest quality mode (high performance)

Definition at line 140 of file ReverbAPI.h.

#### 5.2.4.3 enum **ReverbPresets\_t\_tag**

List of available presets (names explain the preset).

##### Enumerator:

***\_MEDIUM\_HALL*** medium sized concert hall

***\_LARGE\_HALL*** large concert hall

*\_PLATE* rich plate  
*\_CATHEDRAL* large cathedral  
*\_ROOM* small room  
*\_ROOM1* alternative room 1  
*\_ROOM2* alternative room 2

Definition at line 122 of file ReverbAPI.h.

### 5.2.5 Function Documentation

#### 5.2.5.1 `int Reverb_CreateInstance (void ** pCReverb, int iSampleRate, int iNumOfChannels, Quality_t qQualityMode)`

creates an instance of the reverb

**Parameters:**

*pCReverb* : handle  
*iSampleRate* : desired samplerate  
*iNumOfChannels* : desired number of channels  
*qQualityMode* : `_LOW` or `_HIGH` (default), has impact on computing performance vs. quality

**Returns:**

static : 0 if no error occurred

#### 5.2.5.2 `int Reverb_DestroyInstance (void ** pCReverb)`

destroys an instance of the reverb

**Parameters:**

*pCReverb* : handle

**Returns:**

static : 0 if no error occurred

#### 5.2.5.3 `float Reverb_GetBassBoost (void * pCReverb)`

returns the amplify or damping of bass frequencies

**Parameters:**

*pCReverb* : handle

**Returns:**

float : bass boost in dB

**5.2.5.4 float Reverb\_GetDiffusion (void \* *pCReverb*)**

returns the amount of diffusion in the reverberation tail

**Parameters:**

*pCReverb* : handle

**Returns:**

float : diffusion in percent

**5.2.5.5 float Reverb\_GetERGain (void \* *pCReverb*)**

returns the amount of early reflections

**Parameters:**

*pCReverb* : handle

**Returns:**

float : amount of early reflections in dB

**5.2.5.6 float Reverb\_GetERLRBalance (void \* *pCReverb*)**

returns the amount ER/LR balance

**Parameters:**

*pCReverb* : handle

**Returns:**

float : amount of ER/LR balance between -1 and 1

**5.2.5.7 float Reverb\_GetHighCut (void \* *pCReverb*)**

returns the cutoff frequency of a low pass applied to the reverberation

**Parameters:**

*pCReverb* : handle

**Returns:**

float : high cut in Hz

**5.2.5.8 **Hold\_t** Reverb\_GetHold (void \* *pCReverb*)**

returns the hold state

**Parameters:**

*pCReverb* : handle

**Returns:**

**\_Hold\_** : hold state (**\_HOLD\_ON**, **\_HOLD\_OFF**, **\_HOLD\_ACCUMULATE**)

**5.2.5.9 float Reverb\_GetLiveness (void \* *pCReverb*)**

returns the amount of liveness in the reverberation tail

**Parameters:**

*pCReverb* : handle

**Returns:**

float : liveness in percent

**5.2.5.10 float Reverb\_GetLPAmount (void \* *pCReverb*)**

returns the amount of low pass filtering

**Parameters:**

*pCReverb* : handle

**Returns:**

float : low pass amount in percent

**5.2.5.11 float Reverb\_GetLRGain (void \* *pCReverb*)**

returns the amount of late reverberation

**Parameters:**

*pCReverb* : handle

**Returns:**

float : amount of late reverberation in dB

**5.2.5.12 float Reverb\_GetPreDelay (void \* *pCReverb*)**

returns the predelay

**Parameters:**

*pCReverb* : handle

**Returns:**

float : predelay in seconds

**5.2.5.13 **ReverbPresets\_t** Reverb\_GetPreset (void \* *pCReverb*)**

returns the current preset

**Parameters:**

*pCReverb* : handle

**Returns:**

`_ReverbPresets_` : current preset

**5.2.5.14 float Reverb\_GetReverbTime (void \* *pCReverb*)**

returns the reverberation time

**Parameters:**

*pCReverb* : handle

**Returns:**

float : reverberation time in seconds

**5.2.5.15 float Reverb\_GetWetness (void \* *pCReverb*)**

returns the amount of wet (reverberated) signal in output

**Parameters:**

*pCReverb* : handle

**Returns:**

float : wetness in percent

**5.2.5.16 int Reverb\_Process (void \* *pCReverb*, float \* *pInputBuffer*, float \* *pOutputBuffer*, int *iNumOfFrames*)**

add reverb to the input signal

**Parameters:**

*pCReverb* : handle

*pInputBuffer* : buffer with dry input samples

*pOutputBuffer* : buffer for output signal, may be the same as input buffer for inplace processing EXCEPT for the case MONO input / STEREO output. In this case, inplace processing is not allowed

*iNumOfFrames* : number of samples per channel

**Returns:**

void : not `_NO_ERROR` in case of error (e.g. too large input buffer)

**5.2.5.17 void Reverb\_Reset (void \* *pCReverb*)**

resets all internal buffers

**Parameters:**

*pCReverb* : handle

**Returns:**

void :

**5.2.5.18 int Reverb\_SetBassBoost (void \* *pCReverb*, float *fBassBoost*)**

amplify or damp low frequencies in the reverberation. Large rooms often have much longer reverberation time for low frequencies than small ones.

**Parameters:**

*pCReverb* : handle

*fBassBoost* : fBassBoost in dB. Valid parameter range -18...18; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.19 int Reverb\_SetDiffusion (void \* *pCReverb*, float *fDiffusion*)**

set diffusion of the reverberation tail. Large rooms tend to have more diffuse reverberation than small rooms

**Parameters:**

*pCReverb* : handle

*fDiffusion* : fDiffusion in percent; valid parameter range 0...1; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.20 int Reverb\_SetERGain (void \* *pCReverb*, float *fERGainIndB*)**

set the amount of the early reflections. Early reflections will color the reverberation tail

**Parameters:**

*pCReverb* : handle

*fERGainIndB* : fERGainIndB in dB. no parameter range restrictions, however, +-12dB seems to be useful; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.21 int Reverb\_SetERLRBalance (void \* *pCReverb*, float *fERLRBalance*)**

set the amount of ER/LR gain relative to the current ERGain/LRGain-Settings

**Parameters:**

*pCReverb* : handle

*fERLRBalance* : value between -1 (maximum ER, minimum LR) and 1 (minimum ER, maximum LR), default value is 0 changes are dependent on current ERGain/LRGain settings!

**Returns:**

zERROR : not \_NO\_ERROR in case of fERLRBalance outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.22 int Reverb\_SetHighCut (void \* pCReverb, float fLPFreq)**

set the cutoff frequency of a low pass applied to the reverberation. This may be useful for input signals with much high frequency content.

**Parameters:**

*pCReverb* : handle

*fLPFreq* : fLPFreq in dB. Valid parameter range 100...min (18000, Sample-Rate/2.2F); default value depends on selected preset

**Returns:**

int : not \_NO\_ERROR in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.23 int Reverb\_SetHold (void \* pCReverb, Hold\_t eHold)**

experimental/effect function: if hold is set to \_HOLD\_ON, the current reverberation tail is repeated until the function is called with parameter \_HOLD\_OFF. The sound color will change dependent of fLPAmount. If eHold is set to \_HOLD\_ACCUMULATE, the reverb tail will accumulate with the new input signal (THIS COULD LEAD TO CLIPPING VERY FAST!)

**Parameters:**

*pCReverb* : handle

*eHold* : \_HOLD\_ON, \_HOLD\_OFF\_, \_HOLD\_ACCUMULATE

**Returns:**

int :

**5.2.5.24 int Reverb\_SetLiveness (void \* pCReverb, float fLiveness)**

set the liveness of the reverberation tail. High values of liveness could lead to detuning effects in reverberation for tonal input signals.

**Parameters:**

*pCReverb* : handle

*fLiveness* : fLiveness in percent; valid parameter range 0...1; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of `fWetness` outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.25 int Reverb\_SetLPAmount (void \* *pCReverb*, float *fLPAmount*)**

control the amount of low pass filtering in percent; see also `SetHighCut` (.)

**Parameters:**

*pCReverb* : handle

*fLPAmount* : `fLPAmount` in Percent. Valid parameter range 0.0005...0.9995; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of `fWetness` outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.26 int Reverb\_SetLRGain (void \* *pCReverb*, float *fLRGainIndB*)**

set the amount of the late reverberation tail.

**Parameters:**

*pCReverb* : handle

*fLRGainIndB* : `fLRGainIndB` in dB. no parameter range restrictions, however, +-12dB seems to be useful; default value is 0dB

**Returns:**

int : not `_NO_ERROR` in case of `fWetness` outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.27 int Reverb\_SetPreDelay (void \* *pCReverb*, float *fPreDelay*)**

set predelay. The Predelay is an delay inserted before the reverberation tail.

**Parameters:**

*pCReverb* : handle

*fPreDelay* : `fPreDelay` in seconds. Valid parameter range 0...0.3; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of `fWetness` outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.28 int Reverb\_SetPreset (void \* *pCReverb*, **ReverbPresets\_t** *ePreset*)**

Load a preset

**Parameters:**

*pCReverb* : handle

*ePreset* : default value is `_MEDIUM_HALL`

**Returns:**

int :

**5.2.5.29 int Reverb\_SetReverbTime (void \* *pCReverb*, float *fReverbTime*)**

set the reverberation time. Large rooms have a longer reverberation time than small rooms

**Parameters:**

*pCReverb* : handle

*fReverbTime* : fReverbTime in seconds, valid parameter range 1e-10...1000; default value depends on selected preset

**Returns:**

int : not `_NO_ERROR` in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

**5.2.5.30 int Reverb\_SetWetness (void \* *pCReverb*, float *fWetness*)**

set the wetness. If the wetness is set to 0, the dry signal is returned; if the wetness is set to 1, the wet signal (i.e. the reverberation signal) is returned

**Parameters:**

*pCReverb* : handle

*fWetness* : wetness in percent; valid parameter range 0...1; default value: 0.2

**Returns:**

int : not `_NO_ERROR` in case of fWetness outside parameter range, the parameter is then set to the minimum/maximum value

## Index

- `_0_DEGREE_OFF`
  - ChorusAPI.h, 26
  - FlangerAPI.h, 34
  - ResLPAPI.h, 41
  - stereodelayapi.h, 54
- `_180_DEGREE_OFF`
  - ChorusAPI.h, 26
  - FlangerAPI.h, 34
  - ResLPAPI.h, 41
  - stereodelayapi.h, 55
- `_270_DEGREE_OFF`
  - ChorusAPI.h, 26
  - FlangerAPI.h, 34
  - ResLPAPI.h, 41
  - stereodelayapi.h, 55
- `_90_DEGREE_OFF`
  - ChorusAPI.h, 26
  - FlangerAPI.h, 34
  - ResLPAPI.h, 41
  - stereodelayapi.h, 54
- `_CATHEDRAL`
  - ReverbAPI.h, 46
- `_HIGH`
  - ReverbAPI.h, 46
- `_HOLD_ACCUMULATE`
  - ReverbAPI.h, 46
- `_HOLD_OFF`
  - ReverbAPI.h, 46
- `_HOLD_ON`
  - ReverbAPI.h, 46
- `_Hold_`
  - ReverbAPI.h, 46
- `_KILLFILTERAPI_HEADER_-`
  - INCLUDED
  - killfilterapi.h, 38
- `_KillFilter_Modes`
  - killfilterapi.h, 38
- `_LARGE_HALL`
  - ReverbAPI.h, 46
- `_LOW`
  - ReverbAPI.h, 46
- `_LOWEST`
  - ReverbAPI.h, 46
- `_MEDIUM_HALL`
  - ReverbAPI.h, 46
- `_PLATE`
  - ReverbAPI.h, 46
- `_Quality_`
  - ReverbAPI.h, 46
- `__RESLPAPI_HEADER_INCLUDED_`
  - ResLPAPI.h, 40
- `__ROOM`
  - ReverbAPI.h, 46
- `__ReverbPresets_`
  - ReverbAPI.h, 46
- `__AutoPanAPI_HEADER_-`
  - INCLUDED\_\_
  - AutoPanAPI.h, 22
- `__ChorusAPI_HEADER_INCLUDED_-`
  - 
  - ChorusAPI.h, 26
- `__EnvFollowAPI_HEADER_-`
  - INCLUDED\_\_
  - EnvFollowerAPI.h, 31
- `__FLANGERAPI_HEADER_-`
  - INCLUDED\_\_
  - FlangerAPI.h, 34
- `__REVERBAPI_HEADER_-`
  - INCLUDED\_\_
  - ReverbAPI.h, 46
- `__STEREODELAYAPI_HEADER_-`
  - INCLUDED\_\_
  - stereodelayapi.h, 54
- AutoPan\_CreateInstance
  - AutoPanAPI.h, 23
- AutoPan\_DestroyInstance
  - AutoPanAPI.h, 23
- AutoPan\_Process
  - AutoPanAPI.h, 23
- AutoPan\_Reset
  - AutoPanAPI.h, 23
- AutoPan\_SetBPM
  - AutoPanAPI.h, 23
- AutoPan\_SetDepth
  - AutoPanAPI.h, 24
- AutoPan\_SetFreq
  - AutoPanAPI.h, 24
- AutoPan\_SetLFOType
  - AutoPanAPI.h, 24
- AutoPan\_SetPhaseOffset
  - AutoPanAPI.h, 24
- AutoPanAPI.h, 21

- SAW\_LFO, 22
- SINE\_LFO, 22
- TRIANG\_LFO, 22
- AutoPanAPI.h
  - \_\_AutoPanAPI\_HEADER\_-INCLUDED\_\_, 22
  - AutoPan\_CreateInstance, 23
  - AutoPan\_DestroyInstance, 23
  - AutoPan\_Process, 23
  - AutoPan\_Reset, 23
  - AutoPan\_SetBPM, 23
  - AutoPan\_SetDepth, 24
  - AutoPan\_SetFreq, 24
  - AutoPan\_SetLFOType, 24
  - AutoPan\_SetPhaseOffset, 24
  - AutoPanLFOType, 22
- AutoPanLFOType
  - AutoPanAPI.h, 22
- Chorus\_CreateInstance
  - ChorusAPI.h, 26
- Chorus\_DestroyInstance
  - ChorusAPI.h, 27
- Chorus\_Process
  - ChorusAPI.h, 27
- Chorus\_Reset
  - ChorusAPI.h, 27
- Chorus\_SetBlend
  - ChorusAPI.h, 27
- Chorus\_SetBPM
  - ChorusAPI.h, 28
- Chorus\_SetDelay
  - ChorusAPI.h, 28
- Chorus\_SetDepth
  - ChorusAPI.h, 28
- Chorus\_SetFeedback
  - ChorusAPI.h, 28
- Chorus\_SetFeedForward
  - ChorusAPI.h, 29
- Chorus\_SetFreq
  - ChorusAPI.h, 29
- Chorus\_SetLFOType
  - ChorusAPI.h, 29
- Chorus\_SetPhaseOffset
  - ChorusAPI.h, 29
- ChorusAPI.h, 25
  - \_0\_DEGREE\_OFF, 26
  - \_180\_DEGREE\_OFF, 26
  - \_270\_DEGREE\_OFF, 26
  - \_90\_DEGREE\_OFF, 26
- SAW\_LFO, 26
- SINE\_LFO, 26
- TRIANG\_LFO, 26
- ChorusAPI.h
  - \_\_ChorusAPI\_HEADER\_-INCLUDED\_\_, 26
  - Chorus\_CreateInstance, 26
  - Chorus\_DestroyInstance, 27
  - Chorus\_Process, 27
  - Chorus\_Reset, 27
  - Chorus\_SetBlend, 27
  - Chorus\_SetBPM, 28
  - Chorus\_SetDelay, 28
  - Chorus\_SetDepth, 28
  - Chorus\_SetFeedback, 28
  - Chorus\_SetFeedForward, 29
  - Chorus\_SetFreq, 29
  - Chorus\_SetLFOType, 29
  - Chorus\_SetPhaseOffset, 29
  - ChorusLFOType, 26
  - ChorusPhaseOffset, 26
- ChorusLFOType
  - ChorusAPI.h, 26
- ChorusPhaseOffset
  - ChorusAPI.h, 26
- DelayModLFOType
  - stereodelayapi.h, 54
- DelayModPhaseOffset
  - stereodelayapi.h, 54
- docugen.txt, 30
- EnvFollow\_CreateInstance
  - EnvFollowerAPI.h, 31
- EnvFollow\_DestroyInstance
  - EnvFollowerAPI.h, 31
- EnvFollow\_Process
  - EnvFollowerAPI.h, 31
- EnvFollow\_Reset
  - EnvFollowerAPI.h, 32
- EnvFollow\_SetAttack
  - EnvFollowerAPI.h, 32
- EnvFollow\_SetRelease
  - EnvFollowerAPI.h, 32
- EnvFollow\_SetStereoMode
  - EnvFollowerAPI.h, 32
- EnvFollowerAPI.h, 30
  - STEREO\_INDEPENDENT, 31
  - STEREO\_SYNC, 31
- EnvFollowerAPI.h

- \_\_EnvFollowAPI\_HEADER\_-  
INCLUDED\_\_, 31
- EnvFollow\_CreateInstance, 31
- EnvFollow\_DestroyInstance, 31
- EnvFollow\_Process, 31
- EnvFollow\_Reset, 32
- EnvFollow\_SetAttack, 32
- EnvFollow\_SetRelease, 32
- EnvFollow\_SetStereoMode, 32
- EnvFollowStereoMode, 31
- EnvFollowStereoMode  
EnvFollowerAPI.h, 31
- Flanger\_CreateInstance  
FlangerAPI.h, 34
- Flanger\_DestroyInstance  
FlangerAPI.h, 35
- Flanger\_Process  
FlangerAPI.h, 35
- Flanger\_Reset  
FlangerAPI.h, 35
- Flanger\_SetBPM  
FlangerAPI.h, 35
- Flanger\_SetDepth  
FlangerAPI.h, 36
- Flanger\_SetFeedback  
FlangerAPI.h, 36
- Flanger\_SetFreq  
FlangerAPI.h, 36
- Flanger\_SetLFOType  
FlangerAPI.h, 36
- Flanger\_SetPhaseOffset  
FlangerAPI.h, 37
- Flanger\_SetStrength  
FlangerAPI.h, 37
- FlangerAPI.h, 33
- \_0\_DEGREE\_OFF, 34
- \_180\_DEGREE\_OFF, 34
- \_270\_DEGREE\_OFF, 34
- \_90\_DEGREE\_OFF, 34
- SAW\_LFO, 34
- SINE\_LFO, 34
- TRIANG\_LFO, 34
- FlangerAPI.h  
\_\_FLANGERAPI\_HEADER\_-  
INCLUDED\_\_, 34
- Flanger\_CreateInstance, 34
- Flanger\_DestroyInstance, 35
- Flanger\_Process, 35
- Flanger\_Reset, 35
- Flanger\_SetBPM, 35
- Flanger\_SetDepth, 36
- Flanger\_SetFeedback, 36
- Flanger\_SetFreq, 36
- Flanger\_SetLFOType, 36
- Flanger\_SetPhaseOffset, 37
- Flanger\_SetStrength, 37
- FlangerLFOType, 34
- FlangerPhaseOffset, 34
- FlangerLFOType  
FlangerAPI.h, 34
- FlangerPhaseOffset  
FlangerAPI.h, 34
- HIGH\_KILL  
killfilterapi.h, 38
- KillFilter\_CreateInstance  
killfilterapi.h, 38
- KillFilter\_DestroyInstance  
killfilterapi.h, 39
- KillFilter\_Process  
killfilterapi.h, 39
- KillFilter\_Reset  
killfilterapi.h, 39
- KillFilter\_SetMode  
killfilterapi.h, 39
- killfilterapi.h, 37
- \_KILLFILTERAPI\_HEADER\_-  
INCLUDED, 38
- \_KillFilter\_Modes, 38
- HIGH\_KILL, 38
- KillFilter\_CreateInstance, 38
- KillFilter\_DestroyInstance, 39
- KillFilter\_Process, 39
- KillFilter\_Reset, 39
- KillFilter\_SetMode, 39
- LOW\_KILL, 38
- MID\_KILL, 38
- LOW\_KILL  
killfilterapi.h, 38
- LowPass\_CreateInstance  
ResLPAPI.h, 41
- LowPass\_DestroyInstance  
ResLPAPI.h, 41
- LowPass\_Process  
ResLPAPI.h, 42
- LowPass\_Reset  
ResLPAPI.h, 42

- LowPass\_SetBPM
  - ResLPAPI.h, 42
- LowPass\_SetCutOff
  - ResLPAPI.h, 42
- LowPass\_SetFreq
  - ResLPAPI.h, 43
- LowPass\_SetLFOType
  - ResLPAPI.h, 43
- LowPass\_SetMaxMod
  - ResLPAPI.h, 43
- LowPass\_SetPhaseOffset
  - ResLPAPI.h, 43
- LowPass\_SetResonance
  - ResLPAPI.h, 44
- LPMoDLFOType
  - ResLPAPI.h, 41
- LPMoDPhaseOffset
  - ResLPAPI.h, 41
- MID\_KILL
  - killfilterapi.h, 38
- NO\_LFO
  - ResLPAPI.h, 41
  - stereodelayapi.h, 54
- ResLPAPI.h, 40
  - \_0\_DEGREE\_OFF, 41
  - \_180\_DEGREE\_OFF, 41
  - \_270\_DEGREE\_OFF, 41
  - \_90\_DEGREE\_OFF, 41
  - NO\_LFO, 41
  - SAW\_LFO, 41
  - SINE\_LFO, 41
  - TRIANG\_LFO, 41
- ResLPAPI.h
  - \_RESLPAPI\_HEADER\_INCLUDED\_, 40
  - LowPass\_CreateInstance, 41
  - LowPass\_DestroyInstance, 41
  - LowPass\_Process, 42
  - LowPass\_Reset, 42
  - LowPass\_SetBPM, 42
  - LowPass\_SetCutOff, 42
  - LowPass\_SetFreq, 43
  - LowPass\_SetLFOType, 43
  - LowPass\_SetMaxMod, 43
  - LowPass\_SetPhaseOffset, 43
  - LowPass\_SetResonance, 44
  - LPMoDLFOType, 41
  - LPMoDPhaseOffset, 41
  - Reverb\_CreateInstance
    - ReverbAPI.h, 47
  - Reverb\_DestroyInstance
    - ReverbAPI.h, 47
  - Reverb\_GetBassBoost
    - ReverbAPI.h, 47
  - Reverb\_GetDiffusion
    - ReverbAPI.h, 47
  - Reverb\_GetERGain
    - ReverbAPI.h, 47
  - Reverb\_GetHold
    - ReverbAPI.h, 48
  - Reverb\_GetLiveness
    - ReverbAPI.h, 48
  - Reverb\_GetLRGain
    - ReverbAPI.h, 48
  - Reverb\_GetPreDelay
    - ReverbAPI.h, 48
  - Reverb\_GetPreset
    - ReverbAPI.h, 48
  - Reverb\_GetReverbTime
    - ReverbAPI.h, 49
  - Reverb\_GetWetness
    - ReverbAPI.h, 49
  - Reverb\_Process
    - ReverbAPI.h, 49
  - Reverb\_Reset
    - ReverbAPI.h, 49
  - Reverb\_SetBassBoost
    - ReverbAPI.h, 50
  - Reverb\_SetDiffusion
    - ReverbAPI.h, 50
  - Reverb\_SetERGain
    - ReverbAPI.h, 50
  - Reverb\_SetHighCut
    - ReverbAPI.h, 50
  - Reverb\_SetHold
    - ReverbAPI.h, 51
  - Reverb\_SetLiveness
    - ReverbAPI.h, 51
  - Reverb\_SetLPAmount
    - ReverbAPI.h, 51
  - Reverb\_SetLRGain
    - ReverbAPI.h, 52
  - Reverb\_SetPreDelay
    - ReverbAPI.h, 52
  - Reverb\_SetPreset
    - ReverbAPI.h, 52
  - Reverb\_SetReverbTime

- ReverbAPI.h, 52
- Reverb\_SetWetness
  - ReverbAPI.h, 53
- ReverbAPI.h, 44
  - \_CATHEDRAL, 46
  - \_HIGH, 46
  - \_HOLD\_ACCUMULATE, 46
  - \_HOLD\_OFF, 46
  - \_HOLD\_ON, 46
  - \_LARGE\_HALL, 46
  - \_LOW, 46
  - \_LOWEST, 46
  - \_MEDIUM\_HALL, 46
  - \_PLATE, 46
  - \_ROOM, 46
- ReverbAPI.h
  - \_Hold\_, 46
  - \_Quality\_, 46
  - \_ReverbPresets\_, 46
  - \_\_REVERBAPI\_HEADER\_-  
INCLUDED\_\_, 46
  - Reverb\_CreateInstance, 47
  - Reverb\_DestroyInstance, 47
  - Reverb\_GetBassBoost, 47
  - Reverb\_GetDiffusion, 47
  - Reverb\_GetERGain, 47
  - Reverb\_GetHold, 48
  - Reverb\_GetLiveness, 48
  - Reverb\_GetLRGain, 48
  - Reverb\_GetPreDelay, 48
  - Reverb\_GetPreset, 48
  - Reverb\_GetReverbTime, 49
  - Reverb\_GetWetness, 49
  - Reverb\_Process, 49
  - Reverb\_Reset, 49
  - Reverb\_SetBassBoost, 50
  - Reverb\_SetDiffusion, 50
  - Reverb\_SetERGain, 50
  - Reverb\_SetHighCut, 50
  - Reverb\_SetHold, 51
  - Reverb\_SetLiveness, 51
  - Reverb\_SetLPAmount, 51
  - Reverb\_SetLRGain, 52
  - Reverb\_SetPreDelay, 52
  - Reverb\_SetPreset, 52
  - Reverb\_SetReverbTime, 52
  - Reverb\_SetWetness, 53
- SAW\_LFO
  - AutoPanAPI.h, 22
- ChorusAPI.h, 26
- FlangerAPI.h, 34
- ResLPAPI.h, 41
- stereodelayapi.h, 54
- SINE\_LFO
  - AutoPanAPI.h, 22
  - ChorusAPI.h, 26
  - FlangerAPI.h, 34
  - ResLPAPI.h, 41
  - stereodelayapi.h, 54
- STEREO\_INDEPENDENT
  - EnvFollowerAPI.h, 31
- STEREO\_SYNC
  - EnvFollowerAPI.h, 31
- StereoDelay\_CreateInstance
  - stereodelayapi.h, 55
- StereoDelay\_DestroyInstance
  - stereodelayapi.h, 55
- StereoDelay\_Process
  - stereodelayapi.h, 55
- StereoDelay\_Reset
  - stereodelayapi.h, 55
- StereoDelay\_SetBPM
  - stereodelayapi.h, 56
- StereoDelay\_SetDelayTime
  - stereodelayapi.h, 56
- StereoDelay\_SetDepth
  - stereodelayapi.h, 56
- StereoDelay\_SetFeedback
  - stereodelayapi.h, 56
- StereoDelay\_SetFreq
  - stereodelayapi.h, 57
- StereoDelay\_SetLFOType
  - stereodelayapi.h, 57
- StereoDelay\_SetPhaseOffset
  - stereodelayapi.h, 57
- StereoDelay\_SetStereoFade
  - stereodelayapi.h, 57
- StereoDelay\_SetStrength
  - stereodelayapi.h, 58
- stereodelayapi.h, 53
  - \_0\_DEGREE\_OFF, 54
  - \_180\_DEGREE\_OFF, 55
  - \_270\_DEGREE\_OFF, 55
  - \_90\_DEGREE\_OFF, 54
  - \_\_STEREODELAYAPI\_-  
HEADER\_INCLUDED\_\_,  
54
  - DelayModLFOType, 54
  - DelayModPhaseOffset, 54

NO\_LFO, [54](#)  
SAW\_LFO, [54](#)  
SINE\_LFO, [54](#)  
StereoDelay\_CreateInstance, [55](#)  
StereoDelay\_DestroyInstance, [55](#)  
StereoDelay\_Process, [55](#)  
StereoDelay\_Reset, [55](#)  
StereoDelay\_SetBPM, [56](#)  
StereoDelay\_SetDelayTime, [56](#)  
StereoDelay\_SetDepth, [56](#)  
StereoDelay\_SetFeedback, [56](#)  
StereoDelay\_SetFreq, [57](#)  
StereoDelay\_SetLFOType, [57](#)  
StereoDelay\_SetPhaseOffset, [57](#)  
StereoDelay\_SetStereoFade, [57](#)  
StereoDelay\_SetStrength, [58](#)  
TRIANG\_LFO, [54](#)

#### TRIANG\_LFO

AutoPanAPI.h, [22](#)  
ChorusAPI.h, [26](#)  
FlangerAPI.h, [34](#)  
ResLPAPI.h, [41](#)  
stereodelayapi.h, [54](#)