



Delta3D Version 2.4.0

dtAudio::

Reference Manual

Contents

1	Main Page	1
2	Directory Documentation	3
2.1	inc/dtAudio/ Directory Reference	3
2.2	src/dtAudio/ Directory Reference	4
2.3	inc/ Directory Reference	5
2.4	src/ Directory Reference	6
3	Namespace Documentation	7
3.1	dtAudio Namespace Reference	7
3.1.1	Detailed Description	7
3.1.2	Function Documentation	8
3.1.2.1	CheckForError	8
3.1.3	Variable Documentation	8
3.1.3.1	ERROR_CLEARING_STRING	8
4	Class Documentation	9
4.1	AudioActorRegistry Class Reference	9
4.1.1	Constructor & Destructor Documentation	9
4.1.1.1	AudioActorRegistry	9
4.1.2	Member Function Documentation	9
4.1.2.1	RegisterActorTypes	9
4.1.3	Member Data Documentation	9
4.1.3.1	SOUND_ACTOR_TYPE	9
4.2	AudioConfigData Struct Reference	10
4.2.1	Detailed Description	10
4.2.2	Member Enumeration Documentation	10
4.2.2.1	DistanceModel	10
4.2.3	Constructor & Destructor Documentation	10
4.2.3.1	AudioConfigData	10
4.2.4	Member Data Documentation	10
4.2.4.1	distancemodel	10
4.2.4.2	eax	10
4.2.4.3	numSources	10
4.3	AudioManager Class Reference	11
4.3.1	Detailed Description	12
4.3.2	Member Function Documentation	12
4.3.2.1	Config	12

4.3.2.2	Destroy	13
4.3.2.3	FreeSound	13
4.3.2.4	GetContext	13
4.3.2.5	GetDevice	13
4.3.2.6	GetInstance	13
4.3.2.7	GetListener	13
4.3.2.8	GetListenerRelative	13
4.3.2.9	GetSource	13
4.3.2.10	Instantiate	13
4.3.2.11	IsInitialized	13
4.3.2.12	LoadFile	13
4.3.2.13	NewSound	13
4.3.2.14	OnMessage	13
4.3.2.15	SetDistanceModel	14
4.3.2.16	SetDopplerFactor	14
4.3.2.17	SetOpenALDevice	14
4.3.2.18	SetSpeedOfSound	14
4.3.2.19	UnloadFile	14
4.4	FrameData Class Reference	15
4.4.1	Detailed Description	15
4.4.2	Constructor & Destructor Documentation	15
4.4.2.1	FrameData	15
4.4.2.2	FrameData	15
4.4.2.3	~FrameData	15
4.4.3	Friends And Related Function Documentation	15
4.4.3.1	Sound	15
4.5	Listener Class Reference	16
4.5.1	Detailed Description	16
4.5.2	Constructor & Destructor Documentation	16
4.5.2.1	Listener	16
4.5.2.2	~Listener	17
4.5.3	Member Function Documentation	17
4.5.3.1	Clear	17
4.5.3.2	GetGain	17
4.5.3.3	GetVelocity	17
4.5.3.4	OnMessage	17
4.5.3.5	SetGain	17
4.5.3.6	SetVelocity	17
4.6	Sound Class Reference	18
4.6.1	Detailed Description	22
4.6.2	Member Typedef Documentation	22

4.6.2.1	Callback	22
4.6.3	Member Enumeration Documentation	22
4.6.3.1	Command	22
4.6.4	Constructor & Destructor Documentation	23
4.6.4.1	Sound	23
4.6.4.2	~Sound	23
4.6.5	Member Function Documentation	23
4.6.5.1	Clear	23
4.6.5.2	CreateFrameData	23
4.6.5.3	Deserialize	23
4.6.5.4	GetBuffer	23
4.6.5.5	GetDirection	23
4.6.5.6	GetFilename	23
4.6.5.7	GetGain	23
4.6.5.8	GetMaxDistance	23
4.6.5.9	GetMaxGain	23
4.6.5.10	GetMinDistance	23
4.6.5.11	GetMinGain	23
4.6.5.12	GetPitch	24
4.6.5.13	GetPosition	24
4.6.5.14	GetRolloffFactor	24
4.6.5.15	GetSource	24
4.6.5.16	GetState	24
4.6.5.17	GetVelocity	24
4.6.5.18	IsInitialized	24
4.6.5.19	IsListenerRelative	24
4.6.5.20	IsLooping	24
4.6.5.21	IsPaused	24
4.6.5.22	IsPlaying	24
4.6.5.23	IsStopped	24
4.6.5.24	ListenerRelative	24
4.6.5.25	LoadFile	24
4.6.5.26	OnMessage	24
4.6.5.27	Pause	24
4.6.5.28	PauseImmediately	25
4.6.5.29	Play	25
4.6.5.30	PlayImmediately	25
4.6.5.31	ResetState	25
4.6.5.32	Rewind	25
4.6.5.33	RewindImmediately	25
4.6.5.34	RunAllCommandsInQueue	25

4.6.5.35	Serialize	25
4.6.5.36	SetBuffer	25
4.6.5.37	SetDirection	25
4.6.5.38	SetDirectionFromParent	25
4.6.5.39	SetGain	25
4.6.5.40	SetInitialized	25
4.6.5.41	SetListenerRelative	25
4.6.5.42	SetLooping	26
4.6.5.43	SetMaxDistance	26
4.6.5.44	SetMaxGain	26
4.6.5.45	SetMinDistance	26
4.6.5.46	SetMinGain	26
4.6.5.47	SetPitch	26
4.6.5.48	SetPlayCallback	26
4.6.5.49	SetPosition	26
4.6.5.50	SetPositionFromParent	27
4.6.5.51	SetRolloffFactor	27
4.6.5.52	SetSource	27
4.6.5.53	SetState	27
4.6.5.54	SetStopCallback	27
4.6.5.55	SetTransform	27
4.6.5.56	SetVelocity	27
4.6.5.57	Stop	27
4.6.5.58	StopImmediately	27
4.6.5.59	UnloadFile	27
4.6.5.60	UseFrameData	27
4.6.6	Member Data Documentation	28
4.6.6.1	kCommand	28
4.6.6.2	mBuffer	28
4.6.6.3	mCommand	28
4.6.6.4	mCommandState	28
4.6.6.5	mFilename	28
4.6.6.6	mIsInitialized	28
4.6.6.7	mPlayCB	28
4.6.6.8	mPlayCBData	28
4.6.6.9	mSource	28
4.6.6.10	mStopCB	28
4.6.6.11	mStopCBData	28
4.7	SoundActor Class Reference	29
4.7.1	Constructor & Destructor Documentation	29
4.7.1.1	SoundActor	29

4.7.1.2	~SoundActor	29
4.7.2	Member Function Documentation	29
4.7.2.1	GetSound	29
4.7.2.2	GetSound	29
4.8	SoundActorProxy Class Reference	30
4.8.1	Detailed Description	31
4.8.2	Constructor & Destructor Documentation	32
4.8.2.1	SoundActorProxy	32
4.8.2.2	~SoundActorProxy	32
4.8.3	Member Function Documentation	32
4.8.3.1	BuildInvokables	32
4.8.3.2	BuildPropertyMap	32
4.8.3.3	CreateActor	32
4.8.3.4	GetBillBoardIcon	32
4.8.3.5	GetDirection	32
4.8.3.6	GetMaxRandomTime	32
4.8.3.7	GetMinRandomTime	32
4.8.3.8	GetOffsetTime	32
4.8.3.9	GetRenderMode	32
4.8.3.10	GetSound	32
4.8.3.11	GetSound	32
4.8.3.12	GetVelocity	32
4.8.3.13	HandleActorTimers	33
4.8.3.14	IsARandomSoundEffect	33
4.8.3.15	IsPlayedAtStartup	33
4.8.3.16	LoadFile	33
4.8.3.17	OnEnteredWorld	33
4.8.3.18	OnRemovedFromWorld	33
4.8.3.19	Play	33
4.8.3.20	PlayQueued	33
4.8.3.21	SetDirection	33
4.8.3.22	SetMaxRandomTime	33
4.8.3.23	SetMinRandomTime	33
4.8.3.24	SetOffsetTime	33
4.8.3.25	SetPlayAtStartup	33
4.8.3.26	SetToHaveRandomSoundEffect	33
4.8.3.27	SetVelocity	33
4.8.4	Member Data Documentation	34
4.8.4.1	CLASS_NAME	34
4.8.4.2	DEFAULT_RANDOM_TIME_MAX	34
4.8.4.3	DEFAULT_RANDOM_TIME_MIN	34

4.8.4.4	INVOKABLE_TIMER_HANDLER	34
4.8.4.5	PROPERTY_DIRECTION	34
4.8.4.6	PROPERTY_GAIN	34
4.8.4.7	PROPERTY_INITIAL_OFFSET_TIME	34
4.8.4.8	PROPERTY_LISTENER_RELATIVE	34
4.8.4.9	PROPERTY_LOOPING	34
4.8.4.10	PROPERTY_MAX_DISTANCE	34
4.8.4.11	PROPERTY_MAX_GAIN	34
4.8.4.12	PROPERTY_MAX_RANDOM_TIME	34
4.8.4.13	PROPERTY_MIN_GAIN	34
4.8.4.14	PROPERTY_MIN_RANDOM_TIME	34
4.8.4.15	PROPERTY_PITCH	34
4.8.4.16	PROPERTY_PLAY_AS_RANDOM	34
4.8.4.17	PROPERTY_PLAY_AT_STARTUP	34
4.8.4.18	PROPERTY_ROLLOFF_FACTOR	34
4.8.4.19	PROPERTY_SOUND_EFFECT	34
4.8.4.20	PROPERTY_VELOCITY	34
4.8.4.21	TIMER_NAME	34
4.9	SoundCommand Class Reference	35
4.9.1	Detailed Description	35
4.9.2	Constructor & Destructor Documentation	35
4.9.2.1	~SoundCommand	35
4.9.3	Member Data Documentation	35
4.9.3.1	SOUND_COMMAND_PAUSE	35
4.9.3.2	SOUND_COMMAND_PLAY	35
4.9.3.3	SOUND_COMMAND_REWIND	35
4.9.3.4	SOUND_COMMAND_STOP	35
4.10	SoundComponent Class Reference	36
4.10.1	Member Typedef Documentation	38
4.10.1.1	SoundArray	38
4.10.1.2	SoundInfoArray	38
4.10.1.3	SoundInfoMap	38
4.10.1.4	SoundProxyRefArray	38
4.10.2	Constructor & Destructor Documentation	38
4.10.2.1	SoundComponent	38
4.10.2.2	~SoundComponent	38
4.10.3	Member Function Documentation	38
4.10.3.1	AddSound	38
4.10.3.2	AddSound	38
4.10.3.3	AddSound	38
4.10.3.4	AddSoundActorsToWorld	38

4.10.3.5	AddSoundType	38
4.10.3.6	ClearSoundActorArray	39
4.10.3.7	DoSoundCommand	39
4.10.3.8	DoSoundCommand	39
4.10.3.9	DoSoundCommand	39
4.10.3.10	GetAllSounds	39
4.10.3.11	GetSound	39
4.10.3.12	GetSound	39
4.10.3.13	GetSoundActorContainedCount	39
4.10.3.14	GetSoundActorSounds	39
4.10.3.15	GetSoundInfo	40
4.10.3.16	GetSoundInfo	40
4.10.3.17	GetSoundsByType	40
4.10.3.18	OnRemovedFromGM	40
4.10.3.19	Pause	40
4.10.3.20	PauseAllSounds	40
4.10.3.21	PauseAllSoundsByType	40
4.10.3.22	Play	40
4.10.3.23	RemoveAllSounds	40
4.10.3.24	RemoveSound	40
4.10.3.25	RemoveSoundActorsFromWorld	40
4.10.3.26	Rewind	41
4.10.3.27	Stop	41
4.10.3.28	StopAllSounds	41
4.10.3.29	StopAllSoundsByType	41
4.10.4	Member Data Documentation	41
4.10.4.1	DEFAULT_NAME	41
4.11	SoundEffectBinder Class Reference	42
4.11.1	Detailed Description	42
4.11.2	Constructor & Destructor Documentation	42
4.11.2.1	SoundEffectBinder	42
4.11.2.2	~SoundEffectBinder	43
4.11.3	Member Function Documentation	43
4.11.3.1	AddEffectManager	43
4.11.3.2	AddEffectTypeMapping	43
4.11.3.3	AddEffectTypeRange	43
4.11.3.4	Initialize	43
4.11.3.5	RemoveEffectManager	43
4.11.3.6	RemoveEffectTypeMapping	43
4.11.3.7	RemoveEffectTypeRange	43
4.11.3.8	Shutdown	43

4.12	SoundInfo Class Reference	44
4.12.1	Detailed Description	44
4.12.2	Constructor & Destructor Documentation	44
4.12.2.1	SoundInfo	44
4.12.2.2	~SoundInfo	44
4.12.3	Member Function Documentation	44
4.12.3.1	GetSound	44
4.12.3.2	GetSound	44
4.12.3.3	GetType	44
4.13	SoundType Class Reference	45
4.13.1	Detailed Description	45
4.13.2	Constructor & Destructor Documentation	45
4.13.2.1	~SoundType	45
4.13.3	Member Function Documentation	45
4.13.3.1	AddNewType	45
4.13.4	Member Data Documentation	45
4.13.4.1	SOUND_TYPE_DEFAULT	45
4.13.4.2	SOUND_TYPE_MUSIC	45
4.13.4.3	SOUND_TYPE_UI_EFFECT	45
4.13.4.4	SOUND_TYPE_VOICE	45
4.13.4.5	SOUND_TYPE_WORLD_EFFECT	45
5	File Documentation	47
5.1	audioactorregistry.cpp File Reference	47
5.1.1	Function Documentation	47
5.1.1.1	CreatePluginRegistry	47
5.1.1.2	DestroyPluginRegistry	47
5.2	audioactorregistry.h File Reference	48
5.3	audiomanager.cpp File Reference	49
5.3.1	Define Documentation	49
5.3.1.1	BIT	49
5.3.2	Function Documentation	49
5.3.2.1	SetDopplerFactor	49
5.3.2.2	SetSpeedOfSound	49
5.4	audiomanager.h File Reference	50
5.5	dtaudio.h File Reference	51
5.6	export.h File Reference	52
5.6.1	Define Documentation	52
5.6.1.1	DT_AUDIO_EXPORT	52
5.7	listener.cpp File Reference	53
5.8	listener.h File Reference	54

5.9	mainpage.h File Reference	55
5.9.1	Detailed Description	55
5.10	sound.cpp File Reference	56
5.11	sound.h File Reference	57
5.12	soundactorproxy.cpp File Reference	58
5.13	soundactorproxy.h File Reference	59
5.14	soundcommand.cpp File Reference	60
5.15	soundcommand.h File Reference	61
5.16	soundcomponent.cpp File Reference	62
5.17	soundcomponent.h File Reference	63
5.18	soundeffectbinder.cpp File Reference	64
5.19	soundeffectbinder.h File Reference	65
5.20	soundinfo.cpp File Reference	66
5.21	soundinfo.h File Reference	67
5.22	soundtype.cpp File Reference	68
5.23	soundtype.h File Reference	69

Main Page

Delta3D is an Open Source engine which can be used for games, simulations, or other graphical applications.

The Delta3D framework exists as a number of modules, each sitting in its own library, enclosed within its own namespace. At the very core lies the dtCore library. This contains basic, low-level functionality which is mostly required for all 3D applications written in C++.

Around and alongside this sit other supporting libraries, such as dtUtil (containing reusable features which are useful for most applications), dtTerrain (for rendering terrain databases), dtGame, dtNet, etc.

Extensive online documentation is available from the Delta3D Docs section to help in using Delta3D.

The project's original reference guides generated by Doxygen from the source code may be viewed at the Delta3D API Documentation section.

To download source code, binaries, dependencies and sample datasets visit the Delta3D Downloads page.

For more about dependencies see the Delta3D Dependencies page.

The documentation you are looking at can be downloaded from www.3draum.ch.

Enjoy!

Directory Documentation

2.1 inc/dtAudio/ Directory Reference

Files

- file **audioactorregistry.h**
- file **audiomanager.h**
- file **dtaudio.h**
- file **export.h**
- file **listener.h**
- file **mainpage.h**
- file **sound.h**
- file **soundactorproxy.h**
- file **soundcommand.h**
- file **soundcomponent.h**
- file **soundeffectbinder.h**
- file **soundinfo.h**
- file **soundtype.h**

2.2 src/dtAudio/ Directory Reference

Files

- file **audioactorregistry.cpp**
- file **audiomanager.cpp**
- file **listener.cpp**
- file **sound.cpp**
- file **soundactorproxy.cpp**
- file **soundcommand.cpp**
- file **soundcomponent.cpp**
- file **soundeffectbinder.cpp**
- file **soundinfo.cpp**
- file **soundtype.cpp**

2.3 inc/ Directory Reference

Directories

- directory dtAudio

2.4 src/ Directory Reference

Directories

- directory **dtAudio**

Namespace Documentation

3.1 dtAudio Namespace Reference

The Audio Library contains functionality for controlling the audio playback in applications.

Classes

- class **AudioActorRegistry**
- class **AudioManager**
dtAudio::AudioManager (p. 11)
- class **Listener**
dtAudio::Listener (p. 16)
- class **Sound**
dtAudio::Sound (p. 18)
- class **SoundActor**
- class **SoundActorProxy**
*This proxy wraps the **Sound** (p. 18) Delta3D object.*
- class **SoundCommand**
*SOUND COMMAND ENUMERATION CODE Used by the **dtAudio::SoundComponent** (p. 36).*
- class **SoundComponent**
- class **SoundEffectBinder**
A class that binds audible effects to visual effects.
- class **SoundInfo**
SOUND INFO CODE This class creates the relationship between a sound and a sound type.
- class **SoundType**
*SOUND TYPE ENUMERATION CODE Used by the **dtAudio::SoundComponent** (p. 36).*

Functions

- bool **CheckForError** (const std::string &userMessage, const std::string &msgFunction, int lineNumber)

Variables

- const std::string **ERROR_CLEARING_STRING**

3.1.1 Detailed Description

The Audio Library contains functionality for controlling the audio playback in applications.

3.1.2 Function Documentation

3.1.2.1 `bool CheckForError (const std::string & userMessage, const std::string & msgFunction, int lineNumber) [inline]`

3.1.3 Variable Documentation

3.1.3.1 `const std::string ERROR_CLEARING_STRING`

Initial value:

```
"Clearing Error code "  
"system at start of method, if this appears then an error occurred before this  
"  
"method was called."
```

Class Documentation

4.1 AudioActorRegistry Class Reference

```
#include <inc/dtAudio/audioactorregistry.h>
```

Public Member Functions

- **AudioActorRegistry** ()
- virtual void **RegisterActorTypes** ()

Static Public Attributes

- static dtCore::RefPtr< dtDAL::ActorType > **SOUND_ACTOR_TYPE**

4.1.1 Constructor & Destructor Documentation

4.1.1.1 AudioActorRegistry ()

4.1.2 Member Function Documentation

4.1.2.1 void RegisterActorTypes () [virtual]

4.1.3 Member Data Documentation

4.1.3.1 dtCore::RefPtr< dtDAL::ActorType > SOUND_ACTOR_TYPE [static]

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

- **audioactorregistry.h**
- **audioactorregistry.cpp**

4.2 AudioConfigData Struct Reference

Deprecated feb/24/2009 -- setting Distance model with method now,.

```
#include <inc/dtAudio/audiomanager.h>
```

Public Types

- enum **DistanceModel** { **dmNONE** = AL_NONE, **dmINVERSE** = AL_INVERSE_DISTANCE, **dmINVCLAMP** = AL_INVERSE_DISTANCE_CLAMPED }

Public Member Functions

- **AudioConfigData** (unsigned int ns=16, bool ex=false, unsigned int dm=dmINVERSE)

Public Attributes

- unsigned int **distancemodel**
- bool **eax**
- unsigned int **numSources**

4.2.1 Detailed Description

Deprecated feb/24/2009 -- setting Distance model with method now,.

4.2.2 Member Enumeration Documentation

4.2.2.1 enum DistanceModel

Enumerator:

dmNONE

dmINVERSE

dmINVCLAMP

4.2.3 Constructor & Destructor Documentation

4.2.3.1 **AudioConfigData** (unsigned int *ns* = 16, bool *ex* = false, unsigned int *dm* = dmINVERSE)
[inline]

4.2.4 Member Data Documentation

4.2.4.1 unsigned int **distancemodel**

4.2.4.2 bool **eax**

4.2.4.3 unsigned int **numSources**

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

- **audiomanager.h**

4.3 AudioManager Class Reference

dtAudio::AudioManager (p. 11)

```
#include <inc/dtAudio/audiomanager.h>
```

Classes

- struct **BufferData**

BufferData is an internal structure used to identify an OpenAL buffer and hold reference data associated with it.

Public Member Functions

- virtual DEPRECATE_FUNC void **Config** (const **AudioConfigData** &data=**AudioConfigData**())
Deprecated feb/02/2009 All necessary initialization is taken care of.
- void **FreeSound** (**Sound** *sound)
free a sound that the user is finished with
- ALCcontext * **GetContext** ()
*returns the OpenAL context the **AudioManager** (p. 11) is using*
- ALCdevice * **GetDevice** ()
*returns the OpenAL sound device the **AudioManager** (p. 11) is using*
- DEPRECATE_FUNC bool **GetListenerRelative** (**Sound** *sound)
*Deprecated feb/02/2009 in favor of **Sound::IsListenerRelative**() (p. 24).*
- DEPRECATE_FUNC ALuint **GetSource** (**Sound** *sound)
*Deprecated feb/02/2009 in favor of **Sound::GetSource**() (p. 24).*
- ALint **LoadFile** (const std::string &file)
- **Sound** * **NewSound** ()
create or recycle a new sound for the user
- virtual void **OnMessage** (MessageData *data)
receive messages handles the timing messages (pre-post-frame) from the system pushes sounds onto the command queue for later processing
- void **SetDistanceModel** (ALenum dm)
Sets the OpenAL distance model.
- void **SetDopplerFactor** (float f)
Set the OpenAL Doppler factor.
- void **SetOpenALDevice** (const ALCchar *deviceName)
This is an advanced operation that normally isn't necessary! Typically the default device and context are sufficient.
- void **SetSpeedOfSound** (float s)
Set the speed of sound used in OpenAL Doppler calculations.
- bool **UnloadFile** (const std::string &file)
un-load a sound file from a buffer (if use-count is zero)

Static Public Member Functions

- static void **Destroy** ()
destroy the singleton and shutdown OpenAL
- static **AudioManager** & **GetInstance** ()
*access the **AudioManager** (p. 11)*
- static **Listener** * **GetListener** ()
*access the global **Listener** (p. 16)*
- static void **Instantiate** (const std::string &name="audiomanager", ALCdevice *dev=NULL, ALCcontext *cntxt=NULL, bool shutdownPassedInContexts=true)
Create the singleton and initialize OpenAL and ALUT.
- static bool **IsInitialized** ()
Returns true if initialized.

4.3.1 Detailed Description

dtAudio::AudioManager (p. 11) **dtAudio::AudioManager** (p. 11) is the interface to the underlying audio- engine; OpenAL.

Before using, the user must instantiate and configure the **AudioManager** (p. 11):

AudioManager::Instantiate() (p. 13);

Optionally the user can create an **AudioConfigData** (p. 10) structure to pass to the **AudioManager** (p. 11) when configuring set some of the base functionalit of the manager. Currently only the number of sources is set this way. It is encouraged, but not required, that the user know how many channels their audio hardware uses and set the number of sources = the number of channels.

After user is finished with all sound, the **AudioManager** (p. 11) should be freed:

AudioManager::Destroy() (p. 13);

FOR THE USER: Sounds are not created by the user, but they are requested from the **AudioManager** (p. 11). After getting a sound from the **AudioManager** (p. 11) the user then calls the sound's functions. When the user is finished with the sound, the sound should be passed back to the **AudioManager** (p. 11) for free the resource.

There is one global listener which the user also requests from the **AudioManager** (p. 11). After getting the listener from the **AudioManager** (p. 11) the usere then calls the listener's functions. When finished with the listener, the user does NOT free it. The listener is just an interface to AudioManager's protected object and the **AudioManager** (p. 11) will handle it's resources.

In many cases it's more efficeint to preload the sounds into the **AudioManager** (p. 11) before loading them into the individual sounds.

FOR THE DEVELOPER: The **AudioManager** (p. 11) is a repository for all sounds objects and their corresponding buffer.

At pre-frame, the **AudioManager** (p. 11) process all the sounds waiting for command processing. State commands are commnads to change the state of the sound (play, stop, pause, etc.), but not the value of any of the sounds attributes (gain, pitch, etc.). Value commands change the value of a sound's attributes. State commands are pushed onto the Sound's command queue for further processing at the appropriate time (frame change).

At frame time, **AudioManager** (p. 11) process all Sounds with commands in their respective queues.

4.3.2 Member Function Documentation

4.3.2.1 DEPRECATE_FUNC void Config (const AudioConfigData & data = AudioConfigData()) [virtual]

Deprecated feb/02/2009 All necessary intialization is taken care of.

4.3.2.2 void Destroy () [static]

destroy the singleton and shutdown OpenAL

4.3.2.3 void FreeSound (Sound * *sound*)

free a sound that the user is finished with

4.3.2.4 ALCcontext* GetContext () [inline]

returns the OpenAL context the **AudioManager** (p. 11) is using

4.3.2.5 ALCdevice* GetDevice () [inline]

returns the OpenAL sound device the **AudioManager** (p. 11) is using

4.3.2.6 AudioManager & GetInstance () [static]

access the **AudioManager** (p. 11)

4.3.2.7 Listener * GetListener () [static]

access the global **Listener** (p. 16)

4.3.2.8 DEPRECATE_FUNC bool GetListenerRelative (Sound * *sound*)

Deprecated feb/02/2009 in favor of **Sound::IsListenerRelative()** (p. 24).

4.3.2.9 DEPRECATE_FUNC ALuint GetSource (Sound * *sound*)

Deprecated feb/02/2009 in favor of **Sound::GetSource()** (p. 24).

4.3.2.10 void Instantiate (const std::string & *name* = "audiomanager", ALCdevice * *dev* = NULL, ALCcontext * *cntxt* = NULL, bool *shutdownPassedInContexts* = true) [static]

Create the singleton and initialize OpenAL and ALUT. Specifying the OpenAL device and context is considered an advanced, operation. Be sure that you know what you're doing before you try it! Usually you don't need to specify the device and context. Typically, the **AudioManager** (p. 11) creates them automatically.

Note that if you are going to supply a custom device you must ALSO supply a custom context, and vice versa. When you supply the custom device and context, the **AudioManager** (p. 11) assumes that they've been opened, initialized, and that the context has been set to the current OpenAL context.

Parameters

name - The name of the **AudioManager** (p. 11) instance.

dev - The OpenAL device to be used by the **AudioManager** (p. 11).

cntxt - The OpenAL context to be used by the **AudioManager** (p. 11).

shutdownPassedInContexts - this doesn't matter if *cntxt* or *dev* are NULL, but if true, it will close the passed in context when the audio manager destructor is called.

4.3.2.11 static bool IsInitialized () [inline, static]

Returns true if initialized.

4.3.2.12 ALint LoadFile (const std::string & *file*)**4.3.2.13 Sound * NewSound ()**

create or recycle a new sound for the user

4.3.2.14 void OnMessage (MessageData * *data*) [virtual]

receive messages handles the timing messages (pre-post-frame) from the system pushes sounds onto the command queue for later processing

4.3.2.15 void SetDistanceModel (ALenum *dm*)

Sets the OpenAL distance model. Possible parameter values are: AL_INVERSE_DISTANCE, AL_INVERSE_DISTANCE_CLAMPED, AL_LINEAR_DISTANCE, AL_LINEAR_DISTANCE_CLAMPED, AL_EXPONENT_DISTANCE, AL_EXPONENT_DISTANCE_CLAMPED, or AL_NONE

4.3.2.16 void SetDopplerFactor (float *f*)

Set the OpenAL Doppler factor. 0 disables Doppler effect. Values between 0.0 and 1.0 tend to minimize the Doppler effect. 1.0 is the default value. Values greater than 1.0 tend to maximize the Doppler effect. Negative values raise an AL_INVALID_VALUE error.

An very simplified summary of OpenAL calculation for the Doppler effect:

$$\text{shift} = \text{DOPPLER_FACTOR} * \text{freq} * (\text{DOPPLER_VELOCITY} - l.\text{velocity}) / (\text{DOPPLER_VELOCITY} + s.\text{velocity})$$

where l is the listener and s is a sound source.

More detailed calculation specs are in the OpenAL Programmers Guide available at: <http://connect.creativelabs.com/openal>

4.3.2.17 void SetOpenALDevice (const ALCchar * *deviceName*)

This is an advanced operation that normally isn't necessary! Typically the default device and context are sufficient. If you do use this method make sure you brush up on your OpenAL first!

Set the OpenAL device the AudioManager is using. Also sets up an OpenAL context for this device. (Preexisting context and device are closed before doing the preceding)

4.3.2.18 void SetSpeedOfSound (float *s*)

Set the speed of sound used in OpenAL Doppler calculations. Note that OpenAL's default value is 343.3

4.3.2.19 bool UnloadFile (const std::string & *file*)

un-load a sound file from a buffer (if use-count is zero)

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

- **audiomanager.h**
- **audiomanager.cpp**

4.4 FrameData Class Reference

callback function type

```
#include <inc/dtAudio/sound.h>
```

Public Member Functions

- **FrameData** (float *gain*, float *pitch*, unsigned int *playing*=0)
- **FrameData** ()

Protected Member Functions

- **~FrameData** ()

Friends

- class **Sound**
not implemented by design

4.4.1 Detailed Description

callback function type

4.4.2 Constructor & Destructor Documentation

4.4.2.1 FrameData ()

4.4.2.2 FrameData (float *gain*, float *pitch*, unsigned int *playing* = 0)

4.4.2.3 ~FrameData () [protected]

4.4.3 Friends And Related Function Documentation

4.4.3.1 friend class Sound [friend]

not implemented by design

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

- **sound.h**
- **sound.cpp**

4.5 Listener Class Reference

dtAudio::Listener (p. 16)

```
#include <inc/dtAudio/listener.h>
```

Public Member Functions

- **Listener** ()
*Constructor, user does not create directly instead requests the listener from **AudioManager** (p. 11).*
- void **Clear** (void)
clean up listener
- float **GetGain** () const
Returns the master volume of the listener.
- void **GetVelocity** (osg::Vec3f &velocity) const
Get the velocity of the listener.
- virtual void **OnMessage** (MessageData *data)
Message handler's main job is to reposition listener if it's a child of a Transformable in scene-space.
- void **SetGain** (float gain)
Sets the master volume of the listener.
- void **SetVelocity** (const osg::Vec3f &velocity)
Set the velocity of the listener.

Protected Member Functions

- virtual ~**Listener** ()
*Destructor, user does not delete directly **AudioManager** (p. 11) handles destruction.*

4.5.1 Detailed Description

dtAudio::Listener (p. 16) **dtAudio::Listener** (p. 16) is just an interface to the global listener object held within (and protected) by the **dtAudio::AudioManager** (p. 11).

The listener is not usually created directly by the user (new/delete). Instead the user requests the listener from the **AudioManager** (p. 11):

```
Listener* global_ear = AudioManager::GetInstance() (p. 13).GetListener();
```

The user can then call any of the **Listener** (p. 16) interface functions. After the user is finished with the **Listener** (p. 16), there is no need to free it up. The underlying listener object is a global singular which lasts as long as the **AudioManager** (p. 11) exists.

Listener (p. 16) is a transformable, so it can be a child of other transformables (ie. the camera) When a **Listener** (p. 16) is child of another object, it automatically gets positioned in scene-space relative to the parent object every frame, so there is no need to update the Listener's position. The **Listener** (p. 16) position can be set manually in scene-space without having to make it a child of another object, but any position updates must then be made manually.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Listener ()

Constructor, user does not create directly instead requests the listener from **AudioManager** (p. 11).

4.5.2.2 ~Listener () [protected, virtual]

Destructor, user does not delete directly **AudioManager** (p. 11) handles destruction.

4.5.3 Member Function Documentation**4.5.3.1 void Clear (void)**

clean up listener

4.5.3.2 float GetGain (void) const

Returns the master volume of the listener. Returns the current volume

4.5.3.3 void GetVelocity (osg::Vec3f & *velocity*) const

Get the velocity of the listener. Parameters

velocity to get

4.5.3.4 void OnMessage (MessageData * *data*) [virtual]

Message handler's main job is to reposition listener if it's a child of a Transformable in scene-space.

4.5.3.5 void SetGain (float *gain*)

Sets the master volume of the listener. Parameters

gain the new gain

4.5.3.6 void SetVelocity (const osg::Vec3f & *velocity*)

Set the velocity of the listener. Parameters

velocity to set

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

- **listener.h**
- **listener.cpp**

4.6 Sound Class Reference

dtAudio::Sound (p. 18)

```
#include <inc/dtAudio/sound.h>
```

Classes

- class **FrameData**
callback function type

Public Types

- typedef void(* **Callback**)(**Sound** *sound, void *param)
- enum **Command** {
NONE = 0L, **LOAD**, **UNLOAD**, **PLAY**,
PAUSE, **STOP**, **REWIND**, **LOOP**,
UNLOOP, **QUEUE**, **GAIN**, **PITCH**,
POSITION, **DIRECTION**, **VELOCITY**, **ABS**,
REL, **MIN_DIST**, **MAX_DIST**, **ROL_FACT**,
MIN_GAIN, **MAX_GAIN**, **kNumCommands** }

Public Member Functions

- **Sound** ()
Constructor.
- void **Clear** ()
*clean up **Sound** (p. 18) for recycling*
- **FrameData** * **CreateFrameData** () const
Generates and returns a key frame that represents the complete recordable state of this object.
- **FrameData** **Deserialize** (const **FrameData** *data)
Deserializes an XML element representing a state frame, turning it into a new StateFrame instance.
- ALint **GetBuffer** ()
Get this sound's OpenAL buffer ID.
- void **GetDirection** (osg::Vec3 &direction) const
Get the direction of sound.
- const char * **GetFilename** ()
Returns the name of the loaded sound file.
- float **GetGain** () const
Returns the gain of the sound source.
- float **GetMaxDistance** () const
Get the maximum distance that sound plays at min_gain.
- float **GetMaxGain** () const
Get the maximum gain that sound plays at.
- DEPRECATE_FUNC float **GetMinDistance** () const

Deprecated 02/04/2009 -- this method is misleading.

- float **GetMinGain** () const
Get the minimum gain that sound plays at.
- float **GetPitch** () const
Returns the pitch multiplier of the sound source.
- void **GetPosition** (osg::Vec3 &position) const
Get the position of sound.
- float **GetRolloffFactor** () const
Get the rolloff factor describing attenuation curve.
- ALuint **GetSource** ()
*Returns the OpenAL Source ID associated with the loaded Delta3d **Sound** (p. 18) object.*
- void **GetVelocity** (osg::Vec3 &velocity) const
Get the velocity of sound.
- bool **IsInitialized** ()
Get the IsInitialized flag.
- bool **IsListenerRelative** () const
Get the IsListenerRelative flag.
- bool **IsLooping** () const
get the IsLooping flag
- int **IsPaused** () const
Get the IsPaused flag.
- int **IsPlaying** () const
Get the IsPlaying flag.
- int **IsStopped** () const
Get the IsStopped flag.
- DEPRECATE_FUNC void **ListenerRelative** (bool relative)
Deprecated 1/23/2009 in favor of SetListenerRelative.
- void **LoadFile** (const char *file)
Loads the specified sound file.
- void **Pause** ()
*Tells the **AudioManager** (p. 11) to start playing this sound on the next frame step.*
- void **PauseImmediately** ()
*Pauses the sound immediately without reference to the **AudioManager** (p. 11).*
- void **Play** ()
*Tells the **AudioManager** (p. 11) to start playing this on the next frame step.*
- void **PlayImmediately** ()
*Starts playing a sound without reference to the **AudioManager** (p. 11).*

- void **Rewind** ()
*Tells the **AudioManager** (p. 11) to rewind to the beginning of this sound at the next frame step.*
- void **RewindImmediately** ()
*Rewinds a sound immediately without referencing the **AudioManager** (p. 11).*
- void **RunAllCommandsInQueue** ()
Run all commands in the Sound's command queue (also empties the queue).
- XERCES_CPP_NAMESPACE_QUALIFIER DOMElement * **Serialize** (const **FrameData** *d, XERCES_CPP_NAMESPACE_QUALIFIER DOMDocument *doc) const
*turns the **FrameData** (p. 15) into its XML representation.*
- void **SetBuffer** (ALint b)
*Set the Sound's OpenAL buffer ID without going through the **AudioManager** (p. 11).*
- void **SetDirection** (const osg::Vec3 &direction)
Set the direction of sound.
- void **SetDirectionFromParent** ()
- void **SetGain** (float gain)
Sets the gain of the sound source.
- void **SetInitialized** (bool isInit)
Set the IsInitialized flag.
- void **SetListenerRelative** (bool relative)
Flags sound to be relative to listener position.
- void **SetLooping** (bool loop=true)
Sets whether or not to play the sound in a continuous loop.
- void **SetMaxDistance** (float dist)
Sets the distance where there will no longer be any attenuation of the source.
- void **SetMaxGain** (float gain)
Set the maximum gain that sound plays at.
- DEPRECATE_FUNC void **SetMinDistance** (float dist)
Deprecated 02/04/2009 -- this method never did anything.
- void **SetMinGain** (float gain)
Set the minimum gain that sound plays at.
- void **SetPitch** (float pitch)
Sets the pitch multiplier of the sound source.
- virtual void **SetPlayCallback** (**Callback** cb, void *param)
Set callback for when sound starts playing.
- void **SetPosition** (const osg::Vec3 &position)
*Set the position of the **Sound** (p. 18).*
- void **SetPositionFromParent** ()
- void **SetRolloffFactor** (float rolloff)
Set the rolloff factor describing attenuation curve.

- void **SetSource** (ALuint s)
*Sets the OpenAL Source ID associated with the loaded Delta3D **Sound** (p. 18) object.*
- virtual void **SetStopCallback** (**Callback** cb, void *param)
Set callback for when sound stops playing.
- void **SetTransform** (const dtCore::Transform &xform, dtCore::Transformable::CoordSysEnum cs=dtCore::Transformable::ABS_CS)
Set the transform position of sound.
- void **SetVelocity** (const osg::Vec3 &velocity)
Set the velocity of sound.
- void **Stop** ()
Tells the Audio Manager to stop playing the sound at the next frame step.
- void **StopImmediately** ()
*Stops the sound without reference to the **AudioManager** (p. 11).*
- void **UnloadFile** ()
Unloads the specified sound file.
- void **UseFrameData** (const **FrameData** *data)
Used by dtCore::Recorder in playback.

Static Public Attributes

- static const char * **kCommand** [kNumCommands]

Protected Member Functions

- virtual ~**Sound** ()
Destructor.
- unsigned int **GetState** (unsigned int flag) const
- virtual void **OnMessage** (MessageData *data)
Message handler.
- void **ResetState** (unsigned int flag)
- void **SetState** (unsigned int flag)

Protected Attributes

- ALint **mBuffer**
- std::queue< const char * > **mCommand**
- unsigned int **mCommandState**
- std::string **mFilename**
- bool **mIsInitialized**
- **Callback** **mPlayCB**
- void * **mPlayCBData**
- ALuint **mSource**
- **Callback** **mStopCB**
- void * **mStopCBData**

4.6.1 Detailed Description

dtAudio::Sound (p. 18) **dtAudio::Sound** (p. 18) is a little more than just an interface to an object held within (and protected) by the **dtAudio::AudioManager** (p. 11).

Sound (p. 18) objects are not usually created directly by the user (new/delete). Instead the user requests a new sound from the **AudioManager** (p. 11):

```
Sound* mysound = AudioManager::GetInstance() (p. 13).NewSound();
```

The user can then call any of the **Sound** (p. 18) interface functions. After the user is finished with **Sound** (p. 18), it should be returned to the **AudioManager** (p. 11) for recycling:

```
AudioManager::GetInstance() (p. 13).FreeSound( mysound );
```

Sounds do not usually directly call the underlying sound-engine functions, but rather send commands to their command queue. The **AudioManager** (p. 11) will run through all Sounds that have been initialized by it and process their queues at frame time.

Since the **Sound** (p. 18) commands (play, stop, pitch, etc.) may not happen immediately, **Sound** (p. 18) has two callback functions which, if set by the user, will get fired off when the **Sound** (p. 18) actually starts playing and when it actually stops playing.

Sound (p. 18) is a transformable, so it can be a child of other transformables. When a **Sound** (p. 18) is child of another object, it automatically gets positioned in scene-space relative to the parent object every frame, so there is no need to update the Sound's position. The **Sound** (p. 18) position can be set manually in scene-space without having to make it a child of another object, but any position updates must then be made manually.

WARNING ***** The serialization member functions have not been tested. They may work well. Also, the functions require for playback, CreateFrameData and UseFrameData have not been tested. They may be working. An application was not developed to explicitly test these functions due to lack of time.

4.6.2 Member Typedef Documentation

4.6.2.1 typedef void(* Callback)(Sound *sound, void *param)

4.6.3 Member Enumeration Documentation

4.6.3.1 enum Command

Enumerator:

NONE

LOAD

UNLOAD

PLAY

PAUSE

STOP

REWIND

LOOP

UNLOOP

QUEUE

GAIN

PITCH

POSITION

DIRECTION

VELOCITY

ABS

REL

MIN_DIST

MAX_DIST

ROL_FACT
MIN_GAIN
MAX_GAIN
kNumCommands

4.6.4 Constructor & Destructor Documentation

4.6.4.1 Sound ()

Constructor. Typically, user does not create directly instead requests a sound from **AudioManager** (p. 11). The **AudioManager** (p. 11) facilitates sound data buffer management.

4.6.4.2 ~Sound () [protected, virtual]

Destructor. Typically, user does not delete directly instead frees sound to the **AudioManager** (p. 11) so that any buffer(s) that need to be freed also are freed (or not freed... in case other Sounds are using them).

4.6.5 Member Function Documentation

4.6.5.1 void Clear ()

clean up **Sound** (p. 18) for recycling

4.6.5.2 Sound::FrameData * CreateFrameData () const

Generates and returns a key frame that represents the complete recordable state of this object. Returns a new key frame

4.6.5.3 FrameData Deserialize (const FrameData * data)

Deserializes an XML element representing a state frame, turning it into a new StateFrame instance. Parameters

data the element that represents the frame

Returns a newly generated state frame corresponding to the element

4.6.5.4 ALint GetBuffer ()

Get this sound's OpenAL buffer ID.

4.6.5.5 void GetDirection (osg::Vec3 & direction) const

Get the direction of sound. Parameters

direction to get

4.6.5.6 const char* GetFilename () [inline]

Returns the name of the loaded sound file. Returns the name of the loaded file

4.6.5.7 float GetGain () const

Returns the gain of the sound source. Returns the current gain

4.6.5.8 float GetMaxDistance () const

Get the maximum distance that sound plays at min_gain. Returns distance maximum

4.6.5.9 float GetMaxGain () const

Get the maximum gain that sound plays at. Returns gain maximum

4.6.5.10 DEPRECATE_FUNC float GetMinDistance () const

Deprecated 02/04/2009 -- this method is misleading. It is not possible to set a minimum distance via OpenAL.

4.6.5.11 float GetMinGain () const

Get the minimum gain that sound plays at. Returns gain minimum

4.6.5.12 float GetPitch () const

Returns the pitch multiplier of the sound source. Returns the current pitch

4.6.5.13 void GetPosition (osg::Vec3 & *position*) const

Get the position of sound. Parameters

position to get

4.6.5.14 float GetRolloffFactor () const

Get the rolloff factor describing attenuation curve. Returns rolloff factor

4.6.5.15 ALuint GetSource () [inline]

Returns the OpenAL Source ID associated with the loaded Delta3d **Sound** (p. 18) object.

4.6.5.16 unsigned int GetState (unsigned int *flag*) const [inline, protected]**4.6.5.17 void GetVelocity (osg::Vec3 & *velocity*) const**

Get the velocity of sound. Parameters

velocity to get

4.6.5.18 bool IsInitialized () [inline]

Get the IsInitialized flag.

4.6.5.19 bool IsListenerRelative () const

Get the IsListenerRelative flag.

4.6.5.20 bool IsLooping () const

get the IsLooping flag

4.6.5.21 int IsPaused () const [inline]

Get the IsPaused flag.

4.6.5.22 int IsPlaying () const [inline]

Get the IsPlaying flag.

4.6.5.23 int IsStopped () const [inline]

Get the IsStopped flag.

4.6.5.24 DEPRECATE_FUNC void ListenerRelative (bool *relative*) [inline]

Deprecated 1/23/2009 in favor of SetListenerRelative.

4.6.5.25 void LoadFile (const char * *file*)

Loads the specified sound file. Parameters

file the name of the file to load

4.6.5.26 void OnMessage (MessageData * *data*) [protected, virtual]

Message handler. Parameters

data the received message

4.6.5.27 void Pause ()

Tells the **AudioManager** (p. 11) to start playing this sound on the next frame step.

4.6.5.28 void PauseImmediately ()

Pauses the sound immediately without reference to the **AudioManager** (p. 11).

4.6.5.29 void Play ()

Tells the **AudioManager** (p. 11) to start playing this on the next frame step.

4.6.5.30 void PlayImmediately ()

Starts playing a sound without reference to the **AudioManager** (p. 11).

4.6.5.31 void ResetState (unsigned int *flag*) [inline, protected]**4.6.5.32 void Rewind ()**

Tells the **AudioManager** (p. 11) to rewind to the beginning of this sound at the next frame step.

4.6.5.33 void RewindImmediately ()

Rewinds a sound immediately without referencing the **AudioManager** (p. 11).

4.6.5.34 void RunAllCommandsInQueue ()

Run all commands in the Sound's command queue (also empties the queue).

4.6.5.35 DOMElement * Serialize (const FrameData * *d*, XERCES_CPP_NAMESPACE_QUALIFIER DOMDocument * *doc*) const

turns the **FrameData** (p. 15) into its XML representation.

4.6.5.36 void SetBuffer (ALint *b*)

Set the Sound's OpenAL buffer ID without going through the **AudioManager** (p. 11). The typical case is to go through the **AudioManager** (p. 11), but this method is provided for exception cases.

NOTE: This is an advanced operation!! Typically a Sound's OpenAL buffer is set automatically via the **AudioManager** (p. 11). Only tinker with OpenAL buffers directly if you know what you are doing.

4.6.5.37 void SetDirection (const osg::Vec3 & *direction*)

Set the direction of sound. If this is not zero, then the source is automatically considered as "directional" by OpenAL, which means its intensity is not the same in all directions. Check OpenAL specs for details (see AL_DIRECTION).

Parameters

direction to set

4.6.5.38 void SetDirectionFromParent ()**4.6.5.39 void SetGain (float *gain*)**

Sets the gain of the sound source. Parameters

gain the new gain

4.6.5.40 void SetInitialized (bool *isInit*) [inline]

Set the IsInitialized flag.

4.6.5.41 void SetListenerRelative (bool *relative*)

Flags sound to be relative to listener position. IT IS IMPORTANT TO UNDERSTAND EXACTLY WHAT THIS MEANS, otherwise confusion ensues:

When you enable Relative mode on a **Sound** (p. 18) source then its Position, Velocity and Orientation all become relative to the Listener's parameters rather than absolute values.

Therefore: calling this function has no effect on the Listener it ONLY affects the **Sound** (p. 18) source.

You almost never want to set ListenerRelative to be true-- if a **Sound** (p. 18) is in relative mode then when the **Listener** (p. 16) is moved the **Sound** (p. 18) moves with it. For that reason, Delta3D defaults all Sounds to relative = false when Sounds are created.

4.6.5.42 void SetLooping (bool *loop* = true)

Sets whether or not to play the sound in a continuous loop. NOTE that if you set a sound to loop when it is stopped, it will start playing. To prevent this, use the **IsStopped()** (p. 24) method before firing the SetLooping method if you want stopped sounds to stay stopped (you would then need to fire a **Play()** (p. 25) method sometime thereafter to get the stopped looping **Sound** (p. 18) started up).

Parameters

loop : True to play the sound in a loop, otherwise false (plays sound one time then stops)

4.6.5.43 void SetMaxDistance (float *dist*)

Sets the distance where there will no longer be any attenuation of the source. Used with the Inverse Clamped Distance model.

See also dmINVCLAMP

Parameters

dist the maximum distance

4.6.5.44 void SetMaxGain (float *gain*)

Set the maximum gain that sound plays at. Attenuation is clamped to this gain

Parameters

gain set to maximum

4.6.5.45 DEPRECATE_FUNC void SetMinDistance (float *dist*)

Deprecated 02/04/2009 -- this method never did anything. It is not possible to set a minimum distance via OpenAL.

4.6.5.46 void SetMinGain (float *gain*)

Set the minimum gain that sound plays at. Attenuation is clamped to this gain

Parameters

gain set to minimum

4.6.5.47 void SetPitch (float *pitch*)

Sets the pitch multiplier of the sound source. The value is clamped between 0.000001 and 128, but some implementations inexplicably won't take a pitch greater than 2.0, so if this is called with greater than 2.0, the code will attempt to set it, but if fails, it will attempt to set to 2.0.

Parameters

pitch the new pitch

4.6.5.48 void SetPlayCallback (Callback *cb*, void * *param*) [virtual]

Set callback for when sound starts playing. Parameters

cb callback function pointer

param any supplied user data

4.6.5.49 void SetPosition (const osg::Vec3 & *position*)

Set the position of the **Sound** (p. 18). Parameters

position to set

4.6.5.50 void SetPositionFromParent ()**4.6.5.51 void SetRolloffFactor (float *rolloff*)**

Set the rolloff factor describing attenuation curve. Parameters

rolloff factor to set

4.6.5.52 void SetSource (ALuint *s*) [inline]

Sets the OpenAL Source ID associated with the loaded Delta3D **Sound** (p. 18) object. NOTE: This is an advanced operation!! Typically sounds are loaded and sources are set automatically via the **AudioManager** (p. 11). Only tinker with OpenAL sources directly if you know what you are doing.

4.6.5.53 void SetState (unsigned int *flag*) [protected]**4.6.5.54 void SetStopCallback (Callback *cb*, void * *param*) [virtual]**

Set callback for when sound stops playing. Parameters

cb callback function pointer

param any supplied user data

4.6.5.55 void SetTransform (const dtCore::Transform & *xform*, dtCore::Transformable::CoordSysEnum *cs* = dtCore::Transformable::ABS_CS)

Set the transform position of sound. Parameters

**xform* : The new Transform to position this instance

cs : Optional parameter describing the coordinate system of xform Defaults to ABS_CS.

4.6.5.56 void SetVelocity (const osg::Vec3 & *velocity*)

Set the velocity of sound. Parameters

velocity to set

4.6.5.57 void Stop ()

Tells the Audio Manager to stop playing the sound at the next frame step.

4.6.5.58 void StopImmediately ()

Stops the sound without reference to the **AudioManager** (p. 11). Usually you want to go through the **AudioManager** (p. 11) but this method is provided for exceptions to that rule.

4.6.5.59 void UnloadFile ()

Unloads the specified sound file.

4.6.5.60 void UseFrameData (const FrameData * *data*)

Used by dtCore::Recorder in playback. Parameters

data The **Sound::FrameData** (p. 15) containing the Sound's state information.

4.6.6 Member Data Documentation

4.6.6.1 XERCES_CPP_NAMESPACE_USE const char * kCommand [static]

Initial value:

```
{
    "",          "load",      "unload",
    "play",     "pause",    "stop",
    "rewind",   "loop",     "unloop",
    "queue",    "gain",     "pitch",
    "position", "direction", "velocity",
    "absolute", "relative", "mindist",
    "maxdist", "rolloff",  "mingain",
    "maxgain"
}
```

4.6.6.2 ALint mBuffer [protected]

4.6.6.3 std::queue<const char*> mCommand [protected]

4.6.6.4 unsigned int mCommandState [protected]

4.6.6.5 std::string mFilename [protected]

4.6.6.6 bool mIsInitialized [protected]

4.6.6.7 Callback mPlayCB [protected]

4.6.6.8 void* mPlayCBData [protected]

4.6.6.9 ALuint mSource [protected]

4.6.6.10 Callback mStopCB [protected]

4.6.6.11 void* mStopCBData [protected]

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

- sound.h
- sound.cpp

4.7 SoundActor Class Reference

```
#include <inc/dtAudio/soundactorproxy.h>
```

Public Member Functions

- **SoundActor** (dtGame::GameActorProxy &proxy)
- const dtAudio::Sound * **GetSound** () const
- dtAudio::Sound * **GetSound** ()

Protected Member Functions

- virtual ~**SoundActor** ()

4.7.1 Constructor & Destructor Documentation

4.7.1.1 **SoundActor** (dtGame::GameActorProxy & *proxy*)

4.7.1.2 ~**SoundActor** () [protected, virtual]

4.7.2 Member Function Documentation

4.7.2.1 const dtAudio::Sound * **GetSound** () const

4.7.2.2 dtAudio::Sound * **GetSound** ()

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

- soundactorproxy.h
- soundactorproxy.cpp

4.8 SoundActorProxy Class Reference

This proxy wraps the **Sound** (p. 18) Delta3D object.

```
#include <inc/dtAudio/soundactorproxy.h>
```

Public Member Functions

- **SoundActorProxy** ()
Constructor.
- void **BuildInvokables** ()
Builds the invokables of this actor.
- virtual void **BuildPropertyMap** ()
Adds the properties that are common to all Delta3D physical objects.
- virtual dtDAL::ActorProxyIcon * **GetBillBoardIcon** ()
Gets the billboard icon associated with sound actor proxies.
- osg::Vec3 **GetDirection** ()
Gets the direction.
- float **GetMaxRandomTime** () const
- float **GetMinRandomTime** () const
- float **GetOffsetTime** () const
Set the seconds to wait before the initial execution of the sound.
- virtual const ActorProxy::RenderMode & **GetRenderMode** ()
Gets the render mode of sound actor proxies.
- const dtAudio::Sound * **GetSound** () const
*Access the sound object directly without having to grab a hold of the associated **Sound** (p. 18) Actor.*
- dtAudio::Sound * **GetSound** ()
*Access the sound object directly without having to grab a hold of the associated **Sound** (p. 18) Actor.*
- osg::Vec3 **GetVelocity** ()
Gets the velocity.
- void **HandleActorTimers** (const dtGame::Message &msg)
- bool **IsARandomSoundEffect** () const
- bool **IsPlayedAtStartup** () const
Indicates whether or not sound is played at application startup.
- void **LoadFile** (const std::string &fileName)
Loads in a sound file.
- void **OnEnteredWorld** ()
- void **OnRemovedFromWorld** ()
- void **Play** ()
Plays the sound immediately without using timers.
- void **PlayQueued** (float offsetSeconds=0.0f)
Play the sound using the random time range properties set on this sound actor.
- void **SetDirection** (const osg::Vec3 &dir)

Sets the direction.

- void **SetMaxRandomTime** (float seconds)
Set the maximum seconds to wait between random executions of the sound.
- void **SetMinRandomTime** (float seconds)
Set the minimum seconds to wait between random executions of the sound.
- void **SetOffsetTime** (float seconds)
- void **SetPlayAtStartup** (bool val)
Set property that determines if a sound is played at startup.
- void **SetToHaveRandomSoundEffect** (bool value)
- void **SetVelocity** (const osg::Vec3 &vel)
Sets the velocity.

Static Public Attributes

- static const dtUtil::RefString **CLASS_NAME**
- static const float **DEFAULT_RANDOM_TIME_MAX** = 30.0f
- static const float **DEFAULT_RANDOM_TIME_MIN** = 5.0f
- static const dtUtil::RefString **INVOKABLE_TIMER_HANDLER**
- static const dtUtil::RefString **PROPERTY_DIRECTION**
- static const dtUtil::RefString **PROPERTY_GAIN**
- static const dtUtil::RefString **PROPERTY_INITIAL_OFFSET_TIME**
- static const dtUtil::RefString **PROPERTY_LISTENER_RELATIVE**
- static const dtUtil::RefString **PROPERTY_LOOPING**
- static const dtUtil::RefString **PROPERTY_MAX_DISTANCE**
- static const dtUtil::RefString **PROPERTY_MAX_GAIN**
- static const dtUtil::RefString **PROPERTY_MAX_RANDOM_TIME**
- static const dtUtil::RefString **PROPERTY_MIN_GAIN**
- static const dtUtil::RefString **PROPERTY_MIN_RANDOM_TIME**
- static const dtUtil::RefString **PROPERTY_PITCH**
- static const dtUtil::RefString **PROPERTY_PLAY_AS_RANDOM**
- static const dtUtil::RefString **PROPERTY_PLAY_AT_STARTUP**
- static const dtUtil::RefString **PROPERTY_ROLLOFF_FACTOR**
- static const dtUtil::RefString **PROPERTY_SOUND_EFFECT**
- static const dtUtil::RefString **PROPERTY_VELOCITY**
- static const dtUtil::RefString **TIMER_NAME**

Protected Member Functions

- virtual ~**SoundActorProxy** ()
Destructor.
- virtual void **CreateActor** ()
Creates a new positional sound.

4.8.1 Detailed Description

This proxy wraps the **Sound** (p. 18) Delta3D object.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 SoundActorProxy ()

Constructor.

Note You must instantiate, configure, and shutdown the audiomanager in your application ex.

```
dtAudio::AudioManager::Instantiate();
dtAudio::AudioManager::GetManager()->Config(AudioConfigData&)
```

4.8.2.2 ~SoundActorProxy () [protected, virtual]

Destructor.

4.8.3 Member Function Documentation

4.8.3.1 void BuildInvokables ()

Builds the invokables of this actor.

4.8.3.2 void BuildPropertyMap () [virtual]

Adds the properties that are common to all Delta3D physical objects.

4.8.3.3 void CreateActor () [protected, virtual]

Creates a new positional sound.

4.8.3.4 dtDAL::ActorProxyIcon * GetBillBoardIcon () [virtual]

Gets the billboard icon associated with sound actor proxies. Returns

4.8.3.5 osg::Vec3 GetDirection ()

Gets the direction. Returns The current direction

4.8.3.6 float GetMaxRandomTime () const [inline]

Returns Maximum time to wait in whole and/or partial seconds prior to playing the sound.

4.8.3.7 float GetMinRandomTime () const [inline]

Returns Minimum time to wait in whole and/or partial seconds prior to playing the sound.

4.8.3.8 float GetOffsetTime () const [inline]

Set the seconds to wait before the initial execution of the sound. Note that this is only used when the sound actor enters the Game Manager. Parameters

seconds Time to wait in whole and/or partial seconds.

4.8.3.9 virtual const ActorProxy::RenderMode& GetRenderMode () [inline, virtual]

Gets the render mode of sound actor proxies. Returns RenderMode::DRAW_ACTOR_AND_BILLBOARD_ICON. Although this may seem a little strange considering you cannot actually draw a sound, however, this informs the world that this proxy's actor and billboard should be represented in the scene.

4.8.3.10 const dtAudio::Sound * GetSound () const

Access the sound object directly without having to grab a hold of the associated **Sound** (p. 18) Actor. Returns **Sound** (p. 18) object held by the associated **Sound** (p. 18) Actor.

4.8.3.11 dtAudio::Sound * GetSound ()

Access the sound object directly without having to grab a hold of the associated **Sound** (p. 18) Actor. Returns **Sound** (p. 18) object held by the associated **Sound** (p. 18) Actor.

4.8.3.12 osg::Vec3 GetVelocity ()

Gets the velocity. Returns The current velocity

4.8.3.13 void HandleActorTimers (const dtGame::Message & msg)

4.8.3.14 bool IsARandomSoundEffect () const [inline]

4.8.3.15 bool IsPlayedAtStartup () const [inline]

Indicates whether or not sound is played at application startup.

4.8.3.16 void LoadFile (const std::string & fileName)

Loads in a sound file. Parameters

fileName The file to load

4.8.3.17 void OnEnteredWorld ()

4.8.3.18 void OnRemovedFromWorld ()

4.8.3.19 void Play ()

Plays the sound immediately without using timers.

4.8.3.20 void PlayQueued (float offsetSeconds = 0.0f)

Play the sound using the random time range properties set on this sound actor. A sound timer is created in the Game Manager to trigger the sound to play. The sound will play once the timer elapses while this actor is contained in the Game Manager. Parameters

offsetSeconds Time in seconds to wait prior to playing the sound. The offset is added with randomized time if this actor is set as random.

4.8.3.21 void SetDirection (const osg::Vec3 & dir)

Sets the direction. Parameters

dir The direction to Set

4.8.3.22 void SetMaxRandomTime (float seconds) [inline]

Set the maximum seconds to wait between random executions of the sound. Parameters

seconds Maximum time to wait in whole and/or partial seconds.

4.8.3.23 void SetMinRandomTime (float seconds) [inline]

Set the minimum seconds to wait between random executions of the sound. Parameters

seconds Minimum time to wait in whole and/or partial seconds.

4.8.3.24 void SetOffsetTime (float seconds) [inline]

Returns Time to wait in whole and/or partial seconds prior to playing the sound for the first time when the actor enters the Game Manager.

4.8.3.25 void SetPlayAtStartup (bool val) [inline]

Set property that determines if a sound is played at startup.

4.8.3.26 void SetToHaveRandomSoundEffect (bool value) [inline]

4.8.3.27 void SetVelocity (const osg::Vec3 & vel)

Sets the velocity. Parameters

vel The velocity to Set

4.8.4 Member Data Documentation

- 4.8.4.1 `const dtUtil::RefString CLASS_NAME [static]`
- 4.8.4.2 `const float DEFAULT_RANDOM_TIME_MAX = 30.0f [static]`
- 4.8.4.3 `const float DEFAULT_RANDOM_TIME_MIN = 5.0f [static]`
- 4.8.4.4 `const dtUtil::RefString INVOKABLE_TIMER_HANDLER [static]`
- 4.8.4.5 `const dtUtil::RefString PROPERTY_DIRECTION [static]`
- 4.8.4.6 `const dtUtil::RefString PROPERTY_GAIN [static]`
- 4.8.4.7 `const dtUtil::RefString PROPERTY_INITIAL_OFFSET_TIME [static]`
- 4.8.4.8 `const dtUtil::RefString PROPERTY_LISTENER_RELATIVE [static]`
- 4.8.4.9 `const dtUtil::RefString PROPERTY_LOOPING [static]`
- 4.8.4.10 `const dtUtil::RefString PROPERTY_MAX_DISTANCE [static]`
- 4.8.4.11 `const dtUtil::RefString PROPERTY_MAX_GAIN [static]`
- 4.8.4.12 `const dtUtil::RefString PROPERTY_MAX_RANDOM_TIME [static]`
- 4.8.4.13 `const dtUtil::RefString PROPERTY_MIN_GAIN [static]`
- 4.8.4.14 `const dtUtil::RefString PROPERTY_MIN_RANDOM_TIME [static]`
- 4.8.4.15 `const dtUtil::RefString PROPERTY_PITCH [static]`
- 4.8.4.16 `const dtUtil::RefString PROPERTY_PLAY_AS_RANDOM [static]`
- 4.8.4.17 `const dtUtil::RefString PROPERTY_PLAY_AT_STARTUP [static]`
- 4.8.4.18 `const dtUtil::RefString PROPERTY_ROLLOFF_FACTOR [static]`
- 4.8.4.19 `const dtUtil::RefString PROPERTY_SOUND_EFFECT [static]`
- 4.8.4.20 `const dtUtil::RefString PROPERTY_VELOCITY [static]`
- 4.8.4.21 `const dtUtil::RefString TIMER_NAME [static]`

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

- `soundactorproxy.h`
- `soundactorproxy.cpp`

4.9 SoundCommand Class Reference

SOUND COMMAND ENUMERATION CODE Used by the **dtAudio::SoundComponent** (p. 36).

```
#include <inc/dtAudio/soundcommand.h>
```

Static Public Attributes

- static **SoundCommand** SOUND_COMMAND_PAUSE
- static **SoundCommand** SOUND_COMMAND_PLAY
- static **SoundCommand** SOUND_COMMAND_REWIND
- static **SoundCommand** SOUND_COMMAND_STOP

Protected Member Functions

- virtual **~SoundCommand** ()

4.9.1 Detailed Description

SOUND COMMAND ENUMERATION CODE Used by the **dtAudio::SoundComponent** (p. 36).

4.9.2 Constructor & Destructor Documentation

4.9.2.1 **virtual ~SoundCommand** () [*inline, protected, virtual*]

4.9.3 Member Data Documentation

4.9.3.1 **SoundCommand** SOUND_COMMAND_PAUSE [*static*]

4.9.3.2 **SoundCommand** SOUND_COMMAND_PLAY [*static*]

4.9.3.3 **SoundCommand** SOUND_COMMAND_REWIND [*static*]

4.9.3.4 **SoundCommand** SOUND_COMMAND_STOP [*static*]

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

- **soundcommand.h**
- **soundcommand.cpp**

4.10 SoundComponent Class Reference

```
#include <inc/dtAudio/soundcomponent.h>
```

Public Types

- typedef std::vector< **dtAudio::Sound** * > **SoundArray**
- typedef std::vector< **dtAudio::SoundInfo** * > **SoundInfoArray**
- typedef std::map< std::string, dtCore::RefPtr< **dtAudio::SoundInfo** > > **SoundInfoMap**
- typedef std::vector< dtCore::RefPtr< **dtAudio::SoundActorProxy** > > **SoundProxyRefArray**

Public Member Functions

- **SoundComponent** (const std::string &name=DEFAULT_NAME.Get())
- bool **AddSound** (**dtAudio::Sound** &sound, const std::string &soundName, const **dtAudio::SoundType** &soundType=**dtAudio::SoundType::SOUND_TYPE_DEFAULT**)
Register a sound resource by a name and a type.
- **dtAudio::Sound** * **AddSound** (const std::string &soundFile, const std::string &soundName, const **dtAudio::SoundType** &soundType=**dtAudio::SoundType::SOUND_TYPE_DEFAULT**)
Register a sound resource by a name and a type.
- **dtAudio::Sound** * **AddSound** (const std::string &soundFile, const **dtAudio::SoundType** &soundType=**dtAudio::SoundType::SOUND_TYPE_DEFAULT**)
Register a sound resource by a type and use the resource name (soundFile) as the name/handle.
- void **AddSoundActorsToWorld** ()
*Inserts all stored **Sound** (p. 18) Actors back into the Game Manager so they can play their sounds again, triggered by their elapsed timers.*
- bool **AddSoundType** (**dtAudio::SoundType** &newSoundType)
Add a new enumerator for a sound type.
- void **ClearSoundActorArray** ()
*Clear the container that is temporarily holding on to any sound actors that have been evacuated from the world by a call to **RemoveSoundActorsFromWorld**.*
- int **DoSoundCommand** (const **dtAudio::SoundCommand** &command, **SoundArray** &soundArray)
Perform a sound command on a collection of sounds.
- bool **DoSoundCommand** (const **dtAudio::SoundCommand** &command, **dtAudio::Sound** &sound)
Perform an operation on the sound contained in the specified sound.
- bool **DoSoundCommand** (const **dtAudio::SoundCommand** &command, const std::string &soundName)
Perform an operation on a sound.
- void **GetAllSounds** (**SoundArray** &outArray)
Get all sounds registered with this component.
- const **dtAudio::Sound** * **GetSound** (const std::string &soundName) const
- **dtAudio::Sound** * **GetSound** (const std::string &soundName)
Get the sound object mapped to the specified sound name.
- int **GetSoundActorContainedCount** () const
- void **GetSoundActorSounds** (**SoundArray** &outArray)
*Access all sounds held by **Sound** (p. 18) Actors in the Game Manager.*

- const **SoundInfo** * **GetSoundInfo** (const std::string &soundName) const
- **SoundInfo** * **GetSoundInfo** (const std::string &soundName)
Get the sound object that holds the sound object and its type.
- void **GetSoundsByType** (const dtAudio::SoundType &soundType, **SoundArray** &outArray)
Get all sounds registered with this component by type.
- virtual void **OnRemovedFromGM** ()
Called when this component is removed from the game manager.
- bool **Pause** (const std::string &soundName)
Convenience method for pausing a sound by name/handle.
- int **PauseAllSounds** ()
Convenience method.
- int **PauseAllSoundsByType** (const dtAudio::SoundType &soundType)
Convenience method.
- bool **Play** (const std::string &soundName)
Convenience method for playing a sound by name/handle.
- void **RemoveAllSounds** ()
Remove all sounds registered with this component.
- bool **RemoveSound** (const std::string &soundName)
Remove the sound specified by name.
- void **RemoveSoundActorsFromWorld** ()
*Convenience method for removing all **Sound** (p. 18) Actors from the Game Manager.*
- bool **Rewind** (const std::string &soundName)
Convenience method for rewinding a sound by name/handle.
- bool **Stop** (const std::string &soundName)
Convenience method for stopping a sound by name/handle.
- int **StopAllSounds** ()
Convenience method.
- int **StopAllSoundsByType** (const dtAudio::SoundType &soundType)
Convenience method.

Static Public Attributes

- static const dtUtil::RefString **DEFAULT_NAME**

Protected Member Functions

- virtual ~**SoundComponent** ()

4.10.1 Member Typedef Documentation

4.10.1.1 `typedef std::vector<dtAudio::Sound*> SoundArray`

4.10.1.2 `typedef std::vector<dtAudio::SoundInfo*> SoundInfoArray`

4.10.1.3 `typedef std::map<std::string, dtCore::RefPtr<dtAudio::SoundInfo> > SoundInfoMap`

4.10.1.4 `typedef std::vector<dtCore::RefPtr<dtAudio::SoundActorProxy> > SoundProxyRefArray`

4.10.2 Constructor & Destructor Documentation

4.10.2.1 `SoundComponent (const std::string & name = DEFAULT_NAME.Get())`

4.10.2.2 `~SoundComponent () [protected, virtual]`

4.10.3 Member Function Documentation

4.10.3.1 `bool AddSound (dtAudio::Sound & sound, const std::string & soundName, const dtAudio::SoundType & soundType = dtAudio::SoundType::SOUND_TYPE_DEFAULT)`

Register a sound resource by a name and a type. Parameters

sound **Sound** (p. 18) object that is to be added to this component.

soundName Name used as a handle to the sound.

soundType Type of sound being added.

Returns TRUE if the sound was added successfully; FALSE if one of the same name exists.

4.10.3.2 `dtAudio::Sound * AddSound (const std::string & soundFile, const std::string & soundName, const dtAudio::SoundType & soundType = dtAudio::SoundType::SOUND_TYPE_DEFAULT)`

Register a sound resource by a name and a type. Parameters

soundFile Relative path and file name of the sound resource file.

soundName Name used as a handle to the sound.

soundType Type of sound being added.

Returns Pointer to the new sound that was created and added; NULL if not found or one of the same soundName exists.

4.10.3.3 `dtAudio::Sound * AddSound (const std::string & soundFile, const dtAudio::SoundType & soundType = dtAudio::SoundType::SOUND_TYPE_DEFAULT)`

Register a sound resource by a type and use the resource name (soundFile) as the name/handle. Parameters

soundFile Relative path and file name of the sound resource file. that is also used as a handle to the sound.

soundType Type of sound being added.

Returns Pointer to the new sound that was created and added; NULL if not found or one of the same soundName exists.

4.10.3.4 `void AddSoundActorsToWorld ()`

Inserts all stored **Sound** (p. 18) Actors back into the Game Manager so they can play their sounds again, triggered by their elapsed timers. Note that this will remove all the actors from this component's internal list.

4.10.3.5 `bool AddSoundType (dtAudio::SoundType & newSoundType)`

Add a new enumerator for a sound type. This will add the new type to the existing **Sound** (p. 18) Type enumeration only if it does not already exist by the same name as another. Parameters

newSounType Enumerator sound type to add to the existing **Sound** (p. 18) Type enumeration.

Returns TRUE if the enumerator was successfully added.

4.10.3.6 void ClearSoundActorArray ()

Clear the container that is temporarily holding on to any sound actors that have been evacuated from the world by a call to RemoveSoundActorsFromWorld. This will cause the sounds actors to be deleted if no other references point to the sound actors' proxies.

4.10.3.7 int DoSoundCommand (const dtAudio::SoundCommand & *command*, SoundArray & *soundArray*)

Perform a sound command on a collection of sounds. Parameters

command Enumerator used to signify a specific operation to be performed on all the sounds.

soundArray Container with all sounds to be operated on.

Returns Total command successes.

4.10.3.8 bool DoSoundCommand (const dtAudio::SoundCommand & *command*, dtAudio::Sound & *sound*)

Perform an operation on the sound contained in the specified sound. Parameters

command Enumerator used to signify a specific operation to be performed on the sound.

sound **Sound** (p. 18) object to operated on.

Returns TRUE if the sound was found.

4.10.3.9 bool DoSoundCommand (const dtAudio::SoundCommand & *command*, const std::string & *soundName*)

Perform an operation on a sound. Parameters

command Enumerator used to signify a specific operation to be performed on the sound.

soundName Name used as a handle to the sound.

Returns TRUE if the sound was found.

4.10.3.10 void GetAllSounds (SoundArray & *outArray*)

Get all sounds registered with this component. Parameters

outArray Container to capture all sound objects of soundType.

4.10.3.11 const dtAudio::Sound * GetSound (const std::string & *soundName*) const**4.10.3.12 dtAudio::Sound * GetSound (const std::string & *soundName*)**

Get the sound object mapped to the specified sound name. Parameters

soundName Name used as a handle to the sound.

4.10.3.13 int GetSoundActorContainedCount () const

Returns Count of **Sound** (p. 18) Actors current held by this component outside of the Game Manager.

4.10.3.14 void GetSoundActorSounds (SoundArray & *outArray*)

Access all sounds held by **Sound** (p. 18) Actors in the Game Manager. Parameters

outArray Container to capture all sound objects in **Sound** (p. 18) Actors.

4.10.3.15 const dtAudio::SoundInfo * GetSoundInfo (const std::string & *soundName*) const

4.10.3.16 dtAudio::SoundInfo * GetSoundInfo (const std::string & *soundName*)

Get the sound object that holds the sound object and its type. Parameters

soundName Name used as a handle to the sound.

4.10.3.17 void GetSoundsByType (const dtAudio::SoundType & *soundType*, SoundArray & *outArray*)

Get all sounds registered with this component by type. Parameters

soundType Type of sounds to be returned.

outArray Container to capture all sound objects of *soundType*.

4.10.3.18 void OnRemovedFromGM () [virtual]

Called when this component is removed from the game manager.

4.10.3.19 bool Pause (const std::string & *soundName*)

Convenience method for pausing a sound by name/handle. Parameters

soundName Name used as a handle to the sound.

Returns TRUE if the sound was found.

4.10.3.20 int PauseAllSounds ()

Convenience method. Perform a pause operation on all registered sounds. Returns Total command successes.

4.10.3.21 int PauseAllSoundsByType (const dtAudio::SoundType & *soundType*)

Convenience method. Perform a pause operation on all sounds of the specified type. Parameters

soundType Type of sounds to be returned.

Returns Total command successes.

4.10.3.22 bool Play (const std::string & *soundName*)

Convenience method for playing a sound by name/handle. Parameters

soundName Name used as a handle to the sound.

Returns TRUE if the sound was found.

4.10.3.23 void RemoveAllSounds ()

Remove all sounds registered with this component. Note, this does not remove **Sound** (p. 18) Actors from the Game Manager.

4.10.3.24 bool RemoveSound (const std::string & *soundName*)

Remove the sound specified by name. Parameters

soundName Name used as a handle to the sound.

Returns TRUE if the sound was found AND removed.

4.10.3.25 void RemoveSoundActorsFromWorld ()

Convenience method for removing all **Sound** (p. 18) Actors from the Game Manager. This method is an indirect way of pausing all **Sound** (p. 18) Actors in the game. All the actors are maintained by this component so that their resources stay loaded in memory outside of the Game Manager.

4.10.3.26 bool Rewind (const std::string & *soundName*)

Convenience method for rewinding a sound by name/handle. Parameters

soundName Name used as a handle to the sound.

Returns TRUE if the sound was found.

4.10.3.27 bool Stop (const std::string & *soundName*)

Convenience method for stopping a sound by name/handle. Parameters

soundName Name used as a handle to the sound.

Returns TRUE if the sound was found.

4.10.3.28 int StopAllSounds ()

Convenience method. Perform a pause operation on all registered sounds. Returns Total command successes.

4.10.3.29 int StopAllSoundsByType (const dtAudio::SoundType & *soundType*)

Convenience method. Perform a stop operation on all sounds of the specified type. Parameters

soundType Type of sounds to be returned.

Returns Total command successes.

4.10.4 Member Data Documentation**4.10.4.1 const dtUtil::RefString DEFAULT_NAME [static]**

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

- **soundcomponent.h**
- **soundcomponent.cpp**

4.11 SoundEffectBinder Class Reference

A class that binds audible effects to visual effects.

```
#include <inc/dtAudio/soundeffectbinder.h>
```

Classes

- class **SfxObj**
A sound effect object adding mapping management ability to a Delta3D sound object.
- class **SoundEffectListener**

Public Member Functions

- **SoundEffectBinder** (const std::string &name="soundeffectbinder")
Constructor.
- virtual void **AddEffectManager** (dtCore::EffectManager *fxMgr)
Adds an effect manager whos effects we'll monitor.
- virtual void **AddEffectTypeMapping** (const std::string &fxType, const std::string &filename)
Maps the specified effect type to the given filename.
- virtual void **AddEffectTypeRange** (const std::string &fxType, float value, bool minimum_range=true)
Maps the specified effect type to and audible range value.
- virtual void **Initialize** (dtCore::EffectManager *fxMgr=NULL)
*Initialize the **SoundEffectBinder** (p. 42).*
- virtual void **RemoveEffectManager** (dtCore::EffectManager *fxMgr)
Remove an effect manager from our list.
- virtual void **RemoveEffectTypeMapping** (const std::string &fxType)
Removes the specified effect type from the mapping.
- virtual void **RemoveEffectTypeRange** (const std::string &fxType, bool minimum_range=true)
Removes the specified effect type's audible range value.
- virtual void **Shutdown** ()
*Shutdown the **SoundEffectBinder** (p. 42).*

Protected Member Functions

- virtual ~**SoundEffectBinder** ()
Destructor.

4.11.1 Detailed Description

A class that binds audible effects to visual effects. User must specify which (visual) effect manager will be used, then supply a unique id for an audible effect with the sound- filename.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 SoundEffectBinder (const std::string & name = "soundeffectbinder")

Constructor. Parameters

name the instance name

4.11.2.2 ~SoundEffectBinder () [protected, virtual]

Destructor.

4.11.3 Member Function Documentation

4.11.3.1 void AddEffectManager (dtCore::EffectManager * fxMgr) [virtual]

Adds an effect manager whos effects we'll monitor. Parameters

fxMgr the effect manager to add

4.11.3.2 void AddEffectTypeMapping (const std::string & fxType, const std::string & filename) [virtual]

Maps the specified effect type to the given filename. Parameters

fxType the effect type to map

filename the sound filename corresponding to the effect type

4.11.3.3 void AddEffectTypeRange (const std::string & fxType, float value, bool minimum_range = true) [virtual]

Maps the specified effect type to and audible range value. Parameters

fxType the effect type to map

value to map

minimum range if true, else maximum range

4.11.3.4 void Initialize (dtCore::EffectManager * fxMgr = NULL) [virtual]

Initialize the **SoundEffectBinder** (p. 42). Parameters

fxMgr the effect manager to add

4.11.3.5 void RemoveEffectManager (dtCore::EffectManager * fxMgr) [virtual]

Remove an effect manager from our list. Parameters

fxMgr the effect manager to remove

4.11.3.6 void RemoveEffectTypeMapping (const std::string & fxType) [virtual]

Removes the specified effect type from the mapping. Parameters

fxType the effect type to remove

fxType the effect type to map

4.11.3.7 void RemoveEffectTypeRange (const std::string & fxType, bool minimum_range = true) [virtual]

Removes the specified effect type's audible range value. Parameters

fxType the effect type to map

minimum range if true, else maximum range

4.11.3.8 void Shutdown () [virtual]

Shutdown the **SoundEffectBinder** (p. 42).

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

- **soundeffectbinder.h**
- **soundeffectbinder.cpp**

4.12 SoundInfo Class Reference

SOUND INFO CODE This class creates the relationship between a sound and a sound type.

```
#include <inc/dtAudio/soundinfo.h>
```

Public Member Functions

- **SoundInfo** (const dtAudio::SoundType &soundType, dtAudio::Sound &sound)
- const dtAudio::Sound & **GetSound** () const
- dtAudio::Sound & **GetSound** ()
- const dtAudio::SoundType & **GetType** () const

Protected Member Functions

- virtual ~**SoundInfo** ()

4.12.1 Detailed Description

SOUND INFO CODE This class creates the relationship between a sound and a sound type. It also handles play, stop and pause states that have been known to be problems with the **dtAudio** (p. 7) sound system; the system has no way of determining if a sound is playing, which is very bad. Used by the **dtAudio::SoundComponent** (p. 36)

4.12.2 Constructor & Destructor Documentation

4.12.2.1 **SoundInfo** (const dtAudio::SoundType & *soundType*, dtAudio::Sound & *sound*)

4.12.2.2 ~**SoundInfo** () [protected, virtual]

4.12.3 Member Function Documentation

4.12.3.1 const dtAudio::Sound & **GetSound** () const

4.12.3.2 dtAudio::Sound & **GetSound** ()

4.12.3.3 const dtAudio::SoundType & **GetType** () const

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

- **soundinfo.h**
- **soundinfo.cpp**

4.13 SoundType Class Reference

SOUND TYPE ENUMERATION CODE Used by the `dtAudio::SoundComponent` (p. 36).

```
#include <inc/dtAudio/soundtype.h>
```

Static Public Member Functions

- static void **AddNewType** (**SoundType** &soundType)
static

Static Public Attributes

- static **SoundType** SOUND_TYPE_DEFAULT
- static **SoundType** SOUND_TYPE_MUSIC
- static **SoundType** SOUND_TYPE_UI_EFFECT
- static **SoundType** SOUND_TYPE_VOICE
- static **SoundType** SOUND_TYPE_WORLD_EFFECT

Protected Member Functions

- virtual **~SoundType** ()

4.13.1 Detailed Description

SOUND TYPE ENUMERATION CODE Used by the `dtAudio::SoundComponent` (p. 36).

4.13.2 Constructor & Destructor Documentation

4.13.2.1 **virtual ~SoundType** () [*inline, protected, virtual*]

4.13.3 Member Function Documentation

4.13.3.1 **void AddNewType** (**SoundType** & *soundType*) [*static*]
static

4.13.4 Member Data Documentation

4.13.4.1 **SoundType** SOUND_TYPE_DEFAULT [*static*]

4.13.4.2 **SoundType** SOUND_TYPE_MUSIC [*static*]

4.13.4.3 **SoundType** SOUND_TYPE_UI_EFFECT [*static*]

4.13.4.4 **SoundType** SOUND_TYPE_VOICE [*static*]

4.13.4.5 **SoundType** SOUND_TYPE_WORLD_EFFECT [*static*]

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

- **soundtype.h**
- **soundtype.cpp**

File Documentation

5.1 audioactorregistry.cpp File Reference

```
#include <dtAudio/audioactorregistry.h>
```

```
#include <dtAudio/soundactorproxy.h>
```

Functions

- DT_AUDIO_EXPORT dtDAL::ActorPluginRegistry * **CreatePluginRegistry** ()
- DT_AUDIO_EXPORT void **DestroyPluginRegistry** (dtDAL::ActorPluginRegistry *registry)

5.1.1 Function Documentation

5.1.1.1 DT_AUDIO_EXPORT dtDAL::ActorPluginRegistry* **CreatePluginRegistry** ()

5.1.1.2 DT_AUDIO_EXPORT void **DestroyPluginRegistry** (dtDAL::ActorPluginRegistry * *registry*)

5.2 audioactorregistry.h File Reference

```
#include <dtAudio/export.h>
```

```
#include <dtDAL/actorpluginregistry.h>
```

Classes

- class **AudioActorRegistry**

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

5.3 audiomanager.cpp File Reference

```
#include <cassert>
#include <stack>
#include <osg/Vec3>
#include <osg/io_utils>
#include <AL/alut.h>
#include <dtAudio/audiomanager.h>
#include <dtAudio/dtaudio.h>
#include <dtCore/system.h>
#include <dtCore/camera.h>
#include <dtCore/globals.h>
#include <dtUtil/stringutils.h>
#include <iostream>
```

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

Defines

- #define **BIT(a)** (1L<<a)

Functions

- bool **CheckForError** (const std::string &userMessage, const std::string &msgFunction, int lineNumber)
- void **SetDopplerFactor** (float f)
- void **SetSpeedOfSound** (float s)

5.3.1 Define Documentation

5.3.1.1 #define BIT(a) (1L<<a)

5.3.2 Function Documentation

5.3.2.1 void SetDopplerFactor (float f)

5.3.2.2 void SetSpeedOfSound (float s)

5.4 audiomanager.h File Reference

```
#include <cstdint>
#include <vector>
#include <queue>
#include <map>
#include <string>
#include <AL/al.h>
#include <AL/alc.h>
#include <dtCore/base.h>
#include <dtCore/transformable.h>
#include <dtAudio/listener.h>
#include <dtAudio/sound.h>
#include <dtAudio/export.h>
#include <osg/Vec3>
```

Classes

- struct **AudioConfigData**
Deprecated feb/24/2009 -- setting Distance model with method now,.
- class **AudioManager**
dtAudio::AudioManager (p. 11)
- struct **BufferData**
BufferData is an internal structure used to identify an OpenAL buffer and hold reference data associated with it.

Namespaces

- namespace **dtAudio**
The Audio Library contains functionality for controlling the audio playback in applications.

5.5 dtaudio.h File Reference

```
#include <dtAudio/sound.h>
#include <dtAudio/listener.h>
#include <dtAudio/audiomanager.h>
#include <dtAudio/soundeffectbinder.h>
#include <dtUtil/log.h>
```

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

Functions

- bool **CheckForError** (const std::string &userMessage, const std::string &msgFunction, int lineNumber)

Variables

- const std::string **ERROR_CLEARING_STRING**

5.6 export.h File Reference

Defines

- #define `DT_AUDIO_EXPORT`

5.6.1 Define Documentation

5.6.1.1 #define `DT_AUDIO_EXPORT`

5.7 listener.cpp File Reference

```
#include <dtAudio/listener.h>
#include <assert.h>
#include <dtAudio/dtaudio.h>
#include <dtCore/scene.h>
#include <dtCore/system.h>
#include <dtCore/transform.h>
#include <dtCore/collisioncategorydefaults.h>
#include <dtUtil/mathdefines.h>
```

5.8 listener.h File Reference

```
#include <dtCore/transformable.h>
#include <dtAudio/export.h>
#include <osg/Vec3>
```

Classes

- class **Listener**
dtAudio::Listener (p. 16)

Namespaces

- namespace **dtAudio**
The Audio Library contains functionality for controlling the audio playback in applications.

5.9 mainpage.h File Reference

5.9.1 Detailed Description

This file contains Doxygen special commands and text for the **Main Page** (p. ??) and some other minor aspects of this documentation. It is not part of Delta3D.

5.10 sound.cpp File Reference

```
#include <cfloat>
#include <dtAudio/dtaudio.h>
#include <dtAudio/sound.h>
#include <dtCore/scene.h>
#include <dtCore/system.h>
#include <dtCore/transform.h>
#include <dtCore/collisioncategorydefaults.h>
#include <dtUtil/mathdefines.h>
#include <dtUtil/serializer.h>
#include <xercesc/dom/DOMDocument.hpp>
#include <xercesc/dom/DOMElement.hpp>
#include <xercesc/util/XMLString.hpp>
```

5.11 sound.h File Reference

```
#include <queue>
#include <dtCore/transformable.h>
#include <dtAudio/export.h>
#include <AL/alut.h>
#include <osg/Vec3>
#include <xercesc/util/XercesDefs.hpp>
```

Classes

- class **FrameData**
callback function type
- class **Sound**
dtAudio::Sound (p. 18)

Namespaces

- namespace **dtAudio**
The Audio Library contains functionality for controlling the audio playback in applications.

5.12 soundactorproxy.cpp File Reference

```
#include <dtAudio/soundactorproxy.h>
#include <dtDAL/enginepropertytypes.h>
#include <dtDAL/actorproxyicon.h>
#include <dtAudio/audiomanager.h>
#include <dtGame/gamemanager.h>
#include <dtGame/invokable.h>
#include <dtUtil/mathdefines.h>
```

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

5.13 soundactorproxy.h File Reference

```
#include <dtAudio/export.h>
#include <dtDAL/exceptionenum.h>
#include <dtGame/gameactor.h>
```

Classes

- class **SoundActor**
- class **SoundActorProxy**

*This proxy wraps the **Sound** (p. 18) Delta3D object.*

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

5.14 soundcommand.cpp File Reference

```
#include <dtAudio/soundcommand.h>
```

5.15 soundcommand.h File Reference

```
#include <dtAudio/export.h>
```

```
#include <dtUtil/enumeration.h>
```

Classes

- class **SoundCommand**

SOUND COMMAND ENUMERATION CODE Used by the **dtAudio::SoundComponent** (p. 36).

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

5.16 soundcomponent.cpp File Reference

```
#include <dtAudio/soundcomponent.h>
#include <dtAudio/audiomanager.h>
#include <dtAudio/audioactorregistry.h>
#include <dtAudio/soundactorproxy.h>
#include <dtAudio/soundinfo.h>
#include <dtCore/globals.h>
```

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

5.17 soundcomponent.h File Reference

```
#include <dtAudio/export.h>
#include <dtAudio/soundtype.h>
#include <dtAudio/soundcommand.h>
#include <dtGame/gmcomponent.h>
```

Classes

- class **SoundComponent**

Namespaces

- namespace **dtAudio**

The Audio Library contains functionality for controlling the audio playback in applications.

5.18 soundeffectbinder.cpp File Reference

```
#include <dtAudio/soundeffectbinder.h>
#include <cassert>
#include <dtCore/system.h>
#include <dtCore/scene.h>
#include <dtCore/transform.h>
#include <osg/Vec3>
#include <dtUtil/stringutils.h>
```

5.19 soundeffectbinder.h File Reference

```
#include <string>
#include <vector>
#include <map>
#include <queue>
#include <dtCore/base.h>
#include <dtCore/effectmanager.h>
#include <dtAudio/audiomanager.h>
#include <dtAudio/export.h>
#include <dtUtil/functor.h>
```

Classes

- class **SfxObj**
A sound effect object adding mapping management ability to a Delta3D sound object.
- class **SoundEffectBinder**
A class that binds audible effects to visual effects.
- class **SoundEffectListener**

Namespaces

- namespace **dtAudio**
The Audio Library contains functionality for controlling the audio playback in applications.

5.20 soundinfo.cpp File Reference

```
#include <dtAudio/soundinfo.h>
```

5.21 soundinfo.h File Reference

```
#include <dtAudio/export.h>
#include <osg/Referenced>
#include <dtAudio/soundtype.h>
#include <dtAudio/sound.h>
```

Classes

- class **SoundInfo**
SOUND INFO CODE This class creates the relationship between a sound and a sound type.

Namespaces

- namespace **dtAudio**
The Audio Library contains functionality for controlling the audio playback in applications.

5.22 soundtype.cpp File Reference

```
#include <dtAudio/soundtype.h>
```

5.23 soundtype.h File Reference

```
#include <dtUtil/enumeration.h>
```

```
#include <dtAudio/export.h>
```

Classes

- class **SoundType**
*SOUND TYPE ENUMERATION CODE Used by the **dtAudio::SoundComponent** (p. 36).*

Namespaces

- namespace **dtAudio**
The Audio Library contains functionality for controlling the audio playback in applications.

Index

- Symbols -

- ~FrameData
 - dtAudio::Sound::FrameData, 15
- ~Listener
 - dtAudio::Listener, 16
- ~Sound
 - dtAudio::Sound, 23
- ~SoundActor
 - dtAudio::SoundActor, 29
- ~SoundActorProxy
 - dtAudio::SoundActorProxy, 32
- ~SoundCommand
 - dtAudio::SoundCommand, 35
- ~SoundComponent
 - dtAudio::SoundComponent, 38
- ~SoundEffectBinder
 - dtAudio::SoundEffectBinder, 42
- ~SoundInfo
 - dtAudio::SoundInfo, 44
- ~SoundType
 - dtAudio::SoundType, 45

- A -

- ABS
 - dtAudio::Sound, 22
- AddEffectManager
 - dtAudio::SoundEffectBinder, 43
- AddEffectTypeMapping
 - dtAudio::SoundEffectBinder, 43
- AddEffectTypeRange
 - dtAudio::SoundEffectBinder, 43
- AddNewType
 - dtAudio::SoundType, 45
- AddSound
 - dtAudio::SoundComponent, 38
- AddSoundActorsToWorld
 - dtAudio::SoundComponent, 38
- AddSoundType
 - dtAudio::SoundComponent, 38
- AudioActorRegistry
 - dtAudio::AudioActorRegistry, 9
- audioactorregistry.cpp, 47
 - CreatePluginRegistry, 47
 - DestroyPluginRegistry, 47
- audioactorregistry.h, 48
- AudioConfigData, 10
 - AudioConfigData, 10
 - DistanceModel, 10
 - distancemodel, 10
 - dmINVCLAMP, 10
 - dmINVERSE, 10
 - dmNONE, 10
 - eax, 10
 - numSources, 10
- audiomanager.cpp, 49
 - BIT, 49
 - SetDopplerFactor, 49
 - SetSpeedOfSound, 49

audiomanager.h, 50

- B -

- BIT
 - audiomanager.cpp, 49
- BuildInvokables
 - dtAudio::SoundActorProxy, 32
- BuildPropertyMap
 - dtAudio::SoundActorProxy, 32

- C -

- CallBack
 - dtAudio::Sound, 22
- CheckForError
 - dtAudio, 8
- CLASS_NAME
 - dtAudio::SoundActorProxy, 34
- Clear
 - dtAudio::Listener, 17
 - dtAudio::Sound, 23
- ClearSoundActorArray
 - dtAudio::SoundComponent, 38
- Command
 - dtAudio::Sound, 22
- Config
 - dtAudio::AudioManager, 12
- CreateActor
 - dtAudio::SoundActorProxy, 32
- CreateFrameData
 - dtAudio::Sound, 23
- CreatePluginRegistry
 - audioactorregistry.cpp, 47

- D -

- DEFAULT_NAME
 - dtAudio::SoundComponent, 41
- DEFAULT_RANDOM_TIME_MAX
 - dtAudio::SoundActorProxy, 34
- DEFAULT_RANDOM_TIME_MIN
 - dtAudio::SoundActorProxy, 34
- Deserialize
 - dtAudio::Sound, 23
- Destroy
 - dtAudio::AudioManager, 12
- DestroyPluginRegistry
 - audioactorregistry.cpp, 47
- DIRECTION
 - dtAudio::Sound, 22
- DistanceModel
 - AudioConfigData, 10
- distancemodel
 - AudioConfigData, 10
- dmINVCLAMP
 - AudioConfigData, 10
- dmINVERSE
 - AudioConfigData, 10
- dmNONE
 - AudioConfigData, 10
- DoSoundCommand
 - dtAudio::SoundComponent, 39

- DT_AUDIO_EXPORT
 - export.h, 52
- dtAudio, 7
 - CheckForError, 8
 - ERROR_CLEARING_STRING, 8
- dtaudio.h, 51
- dtAudio::AudioActorRegistry, 9
 - AudioActorRegistry, 9
 - RegisterActorTypes, 9
 - SOUND_ACTOR_TYPE, 9
- dtAudio::AudioManager, 11
 - Config, 12
 - Destroy, 12
 - FreeSound, 13
 - GetContext, 13
 - GetDevice, 13
 - GetInstance, 13
 - GetListener, 13
 - GetListenerRelative, 13
 - GetSource, 13
 - Instantiate, 13
 - IsInitialized, 13
 - LoadFile, 13
 - NewSound, 13
 - OnMessage, 13
 - SetDistanceModel, 13
 - SetDopplerFactor, 14
 - SetOpenALDevice, 14
 - SetSpeedOfSound, 14
 - UnloadFile, 14
- dtAudio::Listener, 16
 - ~Listener, 16
 - Clear, 17
 - GetGain, 17
 - GetVelocity, 17
 - Listener, 16
 - OnMessage, 17
 - SetGain, 17
 - SetVelocity, 17
- dtAudio::Sound, 18
 - ~Sound, 23
 - ABS, 22
 - Callback, 22
 - Clear, 23
 - Command, 22
 - CreateFrameData, 23
 - Deserialize, 23
 - DIRECTION, 22
 - GAIN, 22
 - GetBuffer, 23
 - GetDirection, 23
 - GetFilename, 23
 - GetGain, 23
 - GetMaxDistance, 23
 - GetMaxGain, 23
 - GetMinDistance, 23
 - GetMinGain, 23
 - GetPitch, 23
 - GetPosition, 24
 - GetRolloffFactor, 24
 - GetSource, 24
 - GetState, 24
 - GetVelocity, 24
 - IsInitialized, 24
 - IsListenerRelative, 24
 - IsLooping, 24
 - IsPaused, 24
 - IsPlaying, 24
 - IsStopped, 24
 - kCommand, 28
 - kNumCommands, 23
 - ListenerRelative, 24
 - LOAD, 22
 - LoadFile, 24
 - LOOP, 22
 - MAX_DIST, 22
 - MAX_GAIN, 23
 - mBuffer, 28
 - mCommand, 28
 - mCommandState, 28
 - mFilename, 28
 - MIN_DIST, 22
 - MIN_GAIN, 23
 - mIsInitialized, 28
 - mPlayCB, 28
 - mPlayCBData, 28
 - mSource, 28
 - mStopCB, 28
 - mStopCBData, 28
 - NONE, 22
 - OnMessage, 24
 - PAUSE, 22
 - Pause, 24
 - PauseImmediately, 24
 - PITCH, 22
 - PLAY, 22
 - Play, 25
 - PlayImmediately, 25
 - POSITION, 22
 - QUEUE, 22
 - REL, 22
 - ResetState, 25
 - REWIND, 22
 - Rewind, 25
 - RewindImmediately, 25
 - ROL_FACT, 22
 - RunAllCommandsInQueue, 25
 - Serialize, 25
 - SetBuffer, 25
 - SetDirection, 25
 - SetDirectionFromParent, 25
 - SetGain, 25
 - SetInitialized, 25
 - SetListenerRelative, 25
 - SetLooping, 25
 - SetMaxDistance, 26
 - SetMaxGain, 26
 - SetMinDistance, 26
 - SetMinGain, 26
 - SetPitch, 26
 - SetPlayCallback, 26
 - SetPosition, 26
 - SetPositionFromParent, 26
 - SetRolloffFactor, 27
 - SetSource, 27
 - SetState, 27
 - SetStopCallback, 27
 - SetTransform, 27
 - SetVelocity, 27
 - Sound, 23
 - STOP, 22
 - Stop, 27

StopImmediately, 27
 UNLOAD, 22
 UnloadFile, 27
 UNLOOP, 22
 UseFrameData, 27
 VELOCITY, 22
 dtAudio::Sound::FrameData, 15
 ~FrameData, 15
 FrameData, 15
 Sound, 15
 dtAudio::SoundActor, 29
 ~SoundActor, 29
 GetSound, 29
 SoundActor, 29
 dtAudio::SoundActorProxy, 30
 ~SoundActorProxy, 32
 BuildInvokables, 32
 BuildPropertyMap, 32
 CLASS_NAME, 34
 CreateActor, 32
 DEFAULT_RANDOM_TIME_MAX, 34
 DEFAULT_RANDOM_TIME_MIN, 34
 GetBillBoardIcon, 32
 GetDirection, 32
 GetMaxRandomTime, 32
 GetMinRandomTime, 32
 GetOffsetTime, 32
 GetRenderMode, 32
 GetSound, 32
 GetVelocity, 32
 HandleActorTimers, 32
 INVOKABLE_TIMER_HANDLER, 34
 IsARandomSoundEffect, 33
 IsPlayedAtStartup, 33
 LoadFile, 33
 OnEnteredWorld, 33
 OnRemovedFromWorld, 33
 Play, 33
 PlayQueued, 33
 PROPERTY_DIRECTION, 34
 PROPERTY_GAIN, 34
 PROPERTY_INITIAL_OFFSET_TIME, 34
 PROPERTY_LISTENER_RELATIVE, 34
 PROPERTY_LOOPING, 34
 PROPERTY_MAX_DISTANCE, 34
 PROPERTY_MAX_GAIN, 34
 PROPERTY_MAX_RANDOM_TIME, 34
 PROPERTY_MIN_GAIN, 34
 PROPERTY_MIN_RANDOM_TIME, 34
 PROPERTY_PITCH, 34
 PROPERTY_PLAY_AS_RANDOM, 34
 PROPERTY_PLAY_AT_STARTUP, 34
 PROPERTY_ROLLOFF_FACTOR, 34
 PROPERTY_SOUND_EFFECT, 34
 PROPERTY_VELOCITY, 34
 SetDirection, 33
 SetMaxRandomTime, 33
 SetMinRandomTime, 33
 SetOffsetTime, 33
 SetPlayAtStartup, 33
 SetToHaveRandomSoundEffect, 33
 SetVelocity, 33
 SoundActorProxy, 32
 TIMER_NAME, 34
 dtAudio::SoundCommand, 35
 ~SoundCommand, 35
 SOUND_COMMAND_PAUSE, 35
 SOUND_COMMAND_PLAY, 35
 SOUND_COMMAND_REWIND, 35
 SOUND_COMMAND_STOP, 35
 dtAudio::SoundComponent, 36
 ~SoundComponent, 38
 AddSound, 38
 AddSoundActorsToWorld, 38
 AddSoundType, 38
 ClearSoundActorArray, 38
 DEFAULT_NAME, 41
 DoSoundCommand, 39
 GetAllSounds, 39
 GetSound, 39
 GetSoundActorContainedCount, 39
 GetSoundActorSounds, 39
 GetSoundInfo, 39, 40
 GetSoundsByType, 40
 OnRemovedFromGM, 40
 Pause, 40
 PauseAllSounds, 40
 PauseAllSoundsByType, 40
 Play, 40
 RemoveAllSounds, 40
 RemoveSound, 40
 RemoveSoundActorsFromWorld, 40
 Rewind, 40
 SoundArray, 38
 SoundComponent, 38
 SoundInfoArray, 38
 SoundInfoMap, 38
 SoundProxyRefArray, 38
 Stop, 41
 StopAllSounds, 41
 StopAllSoundsByType, 41
 dtAudio::SoundEffectBinder, 42
 ~SoundEffectBinder, 42
 AddEffectManager, 43
 AddEffectTypeMapping, 43
 AddEffectTypeRange, 43
 Initialize, 43
 RemoveEffectManager, 43
 RemoveEffectTypeMapping, 43
 RemoveEffectTypeRange, 43
 Shutdown, 43
 SoundEffectBinder, 42
 dtAudio::SoundInfo, 44
 ~SoundInfo, 44
 GetSound, 44
 GetType, 44
 SoundInfo, 44
 dtAudio::SoundType, 45
 ~SoundType, 45
 AddNewType, 45
 SOUND_TYPE_DEFAULT, 45
 SOUND_TYPE_MUSIC, 45
 SOUND_TYPE_UI_EFFECT, 45
 SOUND_TYPE_VOICE, 45
 SOUND_TYPE_WORLD_EFFECT, 45
- E -
 eax
 AudioConfigData, 10
 ERROR_CLEARING_STRING
 dtAudio, 8
 export.h, 52

- DT_AUDIO_EXPORT, 52
- F -**
- FrameData
 - dtAudio::Sound::FrameData, 15
- FreeSound
 - dtAudio::AudioManager, 13
- G -**
- GAIN
 - dtAudio::Sound, 22
- GetAllSounds
 - dtAudio::SoundComponent, 39
- GetBillBoardIcon
 - dtAudio::SoundActorProxy, 32
- GetBuffer
 - dtAudio::Sound, 23
- GetContext
 - dtAudio::AudioManager, 13
- GetDevice
 - dtAudio::AudioManager, 13
- GetDirection
 - dtAudio::Sound, 23
 - dtAudio::SoundActorProxy, 32
- GetFilename
 - dtAudio::Sound, 23
- GetGain
 - dtAudio::Listener, 17
 - dtAudio::Sound, 23
- GetInstance
 - dtAudio::AudioManager, 13
- GetListener
 - dtAudio::AudioManager, 13
- GetListenerRelative
 - dtAudio::AudioManager, 13
- GetMaxDistance
 - dtAudio::Sound, 23
- GetMaxGain
 - dtAudio::Sound, 23
- GetMaxRandomTime
 - dtAudio::SoundActorProxy, 32
- GetMinDistance
 - dtAudio::Sound, 23
- GetMinGain
 - dtAudio::Sound, 23
- GetMinRandomTime
 - dtAudio::SoundActorProxy, 32
- GetOffsetTime
 - dtAudio::SoundActorProxy, 32
- GetPitch
 - dtAudio::Sound, 23
- GetPosition
 - dtAudio::Sound, 24
- GetRenderMode
 - dtAudio::SoundActorProxy, 32
- GetRolloffFactor
 - dtAudio::Sound, 24
- GetSound
 - dtAudio::SoundActor, 29
 - dtAudio::SoundActorProxy, 32
 - dtAudio::SoundComponent, 39
 - dtAudio::SoundInfo, 44
- GetSoundActorContainedCount
 - dtAudio::SoundComponent, 39
- GetSoundActorSounds
 - dtAudio::SoundComponent, 39
- GetSoundInfo
 - dtAudio::SoundComponent, 39, 40
- GetSoundsByType
 - dtAudio::SoundComponent, 40
- GetSource
 - dtAudio::AudioManager, 13
 - dtAudio::Sound, 24
- GetState
 - dtAudio::Sound, 24
- GetType
 - dtAudio::SoundInfo, 44
- GetVelocity
 - dtAudio::Listener, 17
 - dtAudio::Sound, 24
 - dtAudio::SoundActorProxy, 32
- H -**
- HandleActorTimers
 - dtAudio::SoundActorProxy, 32
- I -**
- inc/ Directory Reference, 5
- inc/dtAudio/ Directory Reference, 3
- Initialize
 - dtAudio::SoundEffectBinder, 43
- Instantiate
 - dtAudio::AudioManager, 13
- INVOKABLE_TIMER_HANDLER
 - dtAudio::SoundActorProxy, 34
- IsARandomSoundEffect
 - dtAudio::SoundActorProxy, 33
- IsInitialized
 - dtAudio::AudioManager, 13
 - dtAudio::Sound, 24
- IsListenerRelative
 - dtAudio::Sound, 24
- IsLooping
 - dtAudio::Sound, 24
- IsPaused
 - dtAudio::Sound, 24
- IsPlayedAtStartup
 - dtAudio::SoundActorProxy, 33
- IsPlaying
 - dtAudio::Sound, 24
- IsStopped
 - dtAudio::Sound, 24
- K -**
- kCommand
 - dtAudio::Sound, 28
- kNumCommands
 - dtAudio::Sound, 23
- L -**
- Listener
 - dtAudio::Listener, 16
- listener.cpp, 53
- listener.h, 54
- ListenerRelative
 - dtAudio::Sound, 24
- LOAD
 - dtAudio::Sound, 22
- LoadFile
 - dtAudio::AudioManager, 13

dtAudio::Sound, 24
 dtAudio::SoundActorProxy, 33
 LOOP
 dtAudio::Sound, 22

- M -

mainpage.h, 55
 MAX_DIST
 dtAudio::Sound, 22
 MAX_GAIN
 dtAudio::Sound, 23
 mBuffer
 dtAudio::Sound, 28
 mCommand
 dtAudio::Sound, 28
 mCommandState
 dtAudio::Sound, 28
 mFilename
 dtAudio::Sound, 28
 MIN_DIST
 dtAudio::Sound, 22
 MIN_GAIN
 dtAudio::Sound, 23
 mIsInitialized
 dtAudio::Sound, 28
 mPlayCB
 dtAudio::Sound, 28
 mPlayCBData
 dtAudio::Sound, 28
 mSource
 dtAudio::Sound, 28
 mStopCB
 dtAudio::Sound, 28
 mStopCBData
 dtAudio::Sound, 28

- N -

NewSound
 dtAudio::AudioManager, 13
 NONE
 dtAudio::Sound, 22
 numSources
 AudioConfigData, 10

- O -

OnEnteredWorld
 dtAudio::SoundActorProxy, 33
 OnMessage
 dtAudio::AudioManager, 13
 dtAudio::Listener, 17
 dtAudio::Sound, 24
 OnRemovedFromGM
 dtAudio::SoundComponent, 40
 OnRemovedFromWorld
 dtAudio::SoundActorProxy, 33

- P -

PAUSE
 dtAudio::Sound, 22
 Pause
 dtAudio::Sound, 24
 dtAudio::SoundComponent, 40
 PauseAllSounds
 dtAudio::SoundComponent, 40
 PauseAllSoundsByType

dtAudio::SoundComponent, 40
 PauseImmediately
 dtAudio::Sound, 24
 PITCH
 dtAudio::Sound, 22
 PLAY
 dtAudio::Sound, 22
 Play
 dtAudio::Sound, 25
 dtAudio::SoundActorProxy, 33
 dtAudio::SoundComponent, 40
 PlayImmediately
 dtAudio::Sound, 25
 PlayQueued
 dtAudio::SoundActorProxy, 33
 POSITION
 dtAudio::Sound, 22
 PROPERTY_DIRECTION
 dtAudio::SoundActorProxy, 34
 PROPERTY_GAIN
 dtAudio::SoundActorProxy, 34
 PROPERTY_INITIAL_OFFSET_TIME
 dtAudio::SoundActorProxy, 34
 PROPERTY_LISTENER_RELATIVE
 dtAudio::SoundActorProxy, 34
 PROPERTY_LOOPING
 dtAudio::SoundActorProxy, 34
 PROPERTY_MAX_DISTANCE
 dtAudio::SoundActorProxy, 34
 PROPERTY_MAX_GAIN
 dtAudio::SoundActorProxy, 34
 PROPERTY_MAX_RANDOM_TIME
 dtAudio::SoundActorProxy, 34
 PROPERTY_MIN_GAIN
 dtAudio::SoundActorProxy, 34
 PROPERTY_MIN_RANDOM_TIME
 dtAudio::SoundActorProxy, 34
 PROPERTY_PITCH
 dtAudio::SoundActorProxy, 34
 PROPERTY_PLAY_AS_RANDOM
 dtAudio::SoundActorProxy, 34
 PROPERTY_PLAY_AT_STARTUP
 dtAudio::SoundActorProxy, 34
 PROPERTY_ROLLOFF_FACTOR
 dtAudio::SoundActorProxy, 34
 PROPERTY_SOUND_EFFECT
 dtAudio::SoundActorProxy, 34
 PROPERTY_VELOCITY
 dtAudio::SoundActorProxy, 34

- Q -

QUEUE
 dtAudio::Sound, 22

- R -

RegisterActorTypes
 dtAudio::AudioActorRegistry, 9
 REL
 dtAudio::Sound, 22
 RemoveAllSounds
 dtAudio::SoundComponent, 40
 RemoveEffectManager
 dtAudio::SoundEffectBinder, 43
 RemoveEffectTypeMapping
 dtAudio::SoundEffectBinder, 43

- RemoveEffectTypeRange
 - dtAudio::SoundEffectBinder, 43
- RemoveSound
 - dtAudio::SoundComponent, 40
- RemoveSoundActorsFromWorld
 - dtAudio::SoundComponent, 40
- ResetState
 - dtAudio::Sound, 25
- REWIND
 - dtAudio::Sound, 22
- Rewind
 - dtAudio::Sound, 25
 - dtAudio::SoundComponent, 40
- RewindImmediately
 - dtAudio::Sound, 25
- ROL_FACT
 - dtAudio::Sound, 22
- RunAllCommandsInQueue
 - dtAudio::Sound, 25
- S -**
- Serialize
 - dtAudio::Sound, 25
- SetBuffer
 - dtAudio::Sound, 25
- SetDirection
 - dtAudio::Sound, 25
 - dtAudio::SoundActorProxy, 33
- SetDirectionFromParent
 - dtAudio::Sound, 25
- SetDistanceModel
 - dtAudio::AudioManager, 13
- SetDopplerFactor
 - audiomanager.cpp, 49
 - dtAudio::AudioManager, 14
- SetGain
 - dtAudio::Listener, 17
 - dtAudio::Sound, 25
- SetInitialized
 - dtAudio::Sound, 25
- SetListenerRelative
 - dtAudio::Sound, 25
- SetLooping
 - dtAudio::Sound, 25
- SetMaxDistance
 - dtAudio::Sound, 26
- SetMaxGain
 - dtAudio::Sound, 26
- SetMaxRandomTime
 - dtAudio::SoundActorProxy, 33
- SetMinDistance
 - dtAudio::Sound, 26
- SetMinGain
 - dtAudio::Sound, 26
- SetMinRandomTime
 - dtAudio::SoundActorProxy, 33
- SetOffsetTime
 - dtAudio::SoundActorProxy, 33
- SetOpenALDevice
 - dtAudio::AudioManager, 14
- SetPitch
 - dtAudio::Sound, 26
- SetPlayAtStartup
 - dtAudio::SoundActorProxy, 33
- SetPlayCallback
 - dtAudio::Sound, 26
- SetPosition
 - dtAudio::Sound, 26
- SetPositionFromParent
 - dtAudio::Sound, 26
- SetRolloffFactor
 - dtAudio::Sound, 27
- SetSource
 - dtAudio::Sound, 27
- SetSpeedOfSound
 - audiomanager.cpp, 49
 - dtAudio::AudioManager, 14
- SetState
 - dtAudio::Sound, 27
- SetStopCallback
 - dtAudio::Sound, 27
- SetToHaveRandomSoundEffect
 - dtAudio::SoundActorProxy, 33
- SetTransform
 - dtAudio::Sound, 27
- SetVelocity
 - dtAudio::Listener, 17
 - dtAudio::Sound, 27
 - dtAudio::SoundActorProxy, 33
- Shutdown
 - dtAudio::SoundEffectBinder, 43
- Sound
 - dtAudio::Sound, 23
 - dtAudio::Sound::FrameData, 15
- sound.cpp, 56
- sound.h, 57
- SOUND_ACTOR_TYPE
 - dtAudio::AudioActorRegistry, 9
- SOUND_COMMAND_PAUSE
 - dtAudio::SoundCommand, 35
- SOUND_COMMAND_PLAY
 - dtAudio::SoundCommand, 35
- SOUND_COMMAND_REWIND
 - dtAudio::SoundCommand, 35
- SOUND_COMMAND_STOP
 - dtAudio::SoundCommand, 35
- SOUND_TYPE_DEFAULT
 - dtAudio::SoundType, 45
- SOUND_TYPE_MUSIC
 - dtAudio::SoundType, 45
- SOUND_TYPE_UI_EFFECT
 - dtAudio::SoundType, 45
- SOUND_TYPE_VOICE
 - dtAudio::SoundType, 45
- SOUND_TYPE_WORLD_EFFECT
 - dtAudio::SoundType, 45
- SoundActor
 - dtAudio::SoundActor, 29
- SoundActorProxy
 - dtAudio::SoundActorProxy, 32
- soundactorproxy.cpp, 58
- soundactorproxy.h, 59
- SoundArray
 - dtAudio::SoundComponent, 38
- soundcommand.cpp, 60
- soundcommand.h, 61
- SoundComponent
 - dtAudio::SoundComponent, 38
- soundcomponent.cpp, 62
- soundcomponent.h, 63
- SoundEffectBinder
 - dtAudio::SoundEffectBinder, 42

soundeffectbinder.cpp, 64
soundeffectbinder.h, 65
SoundInfo
 dtAudio::SoundInfo, 44
soundinfo.cpp, 66
soundinfo.h, 67
SoundInfoArray
 dtAudio::SoundComponent, 38
SoundInfoMap
 dtAudio::SoundComponent, 38
SoundProxyRefArray
 dtAudio::SoundComponent, 38
soundtype.cpp, 68
soundtype.h, 69
src/ Directory Reference, 6
src/dtAudio/ Directory Reference, 4
STOP
 dtAudio::Sound, 22
Stop
 dtAudio::Sound, 27
 dtAudio::SoundComponent, 41
StopAllSounds
 dtAudio::SoundComponent, 41
StopAllSoundsByType
 dtAudio::SoundComponent, 41
StopImmediately
 dtAudio::Sound, 27

- T -

TIMER_NAME
 dtAudio::SoundActorProxy, 34

- U -

UNLOAD
 dtAudio::Sound, 22
UnloadFile
 dtAudio::AudioManager, 14
 dtAudio::Sound, 27
UNLOOP
 dtAudio::Sound, 22
UseFrameData
 dtAudio::Sound, 27

- V -

VELOCITY
 dtAudio::Sound, 22