

ObjectAL

Generated by Doxygen 1.7.2

Sat Feb 19 2011 12:35:40



# Contents

<b>1</b>	<b>ObjectAL for iPhone</b>	<b>1</b>
1.1	Contents	1
1.2	Introduction	1
1.3	ObjectAL and OpenAL	2
1.4	Adding ObjectAL to your project	4
1.4.1	Installing the ObjectAL Documentation into XCode	4
1.5	Compile-Time Configuration	5
1.6	Audio Formats	5
1.6.1	OALAudioTrack Supported Formats	5
1.6.2	OpenAL Supported Formats	5
1.7	Choosing Playback Types	5
1.8	Using OALSimpleAudio	6
1.9	Using the OpenAL Objects and OALAudioTrack	8
1.10	Other Examples	10
1.11	iOS Issues that can impede playback	11
1.11.1	MPMoviePlayerController on iOS 3.x	11
1.11.2	MPMusicPlayerController on iOS 4.0	12
1.12	Simulator Issues	12
1.12.1	Simulator Limitations	12
1.12.2	Error Codes on the Simulator	12
1.12.3	Playback Issues	12
1.12.4	No OpenAL Sound in Simulator	13
1.12.5	Simulator Freezups	13
<b>2</b>	<b>Class Index</b>	<b>15</b>
2.1	Class Hierarchy	15
<b>3</b>	<b>Class Index</b>	<b>17</b>
3.1	Class List	17
<b>4</b>	<b>Class Documentation</b>	<b>21</b>
4.1	ALBuffer Class Reference	21
4.1.1	Detailed Description	22
4.1.2	Member Function Documentation	22
4.1.2.1	bufferWithName:data:size:format:frequency:	22
4.1.2.2	initWithName:data:size:format:frequency:	23
4.1.3	Member Data Documentation	23
4.1.3.1	bufferData	23
4.1.4	Property Documentation	23

4.1.4.1	bits	23
4.1.4.2	bufferId	23
4.1.4.3	channels	24
4.1.4.4	device	24
4.1.4.5	duration	24
4.1.4.6	format	24
4.1.4.7	freeDataOnDestroy	24
4.1.4.8	frequency	24
4.1.4.9	name	24
4.1.4.10	size	24
4.2	ALCaptureDevice Class Reference	24
4.2.1	Detailed Description	26
4.2.2	Member Function Documentation	26
4.2.2.1	close	26
4.2.2.2	closeOSResources	26
4.2.2.3	deviceWithDeviceSpecifier:frequency:format:bufferSize:	26
4.2.2.4	getProcAddress:	27
4.2.2.5	initWithDeviceSpecifier:frequency:format:bufferSize:	27
4.2.2.6	isExtensionPresent:	27
4.2.2.7	moveSamples:toBuffer:	27
4.2.2.8	startCapture	28
4.2.2.9	stopCapture	28
4.2.3	Property Documentation	28
4.2.3.1	captureSamples	28
4.2.3.2	device	28
4.2.3.3	extensions	28
4.2.3.4	majorVersion	28
4.2.3.5	minorVersion	29
4.3	ALChannelSource Class Reference	29
4.3.1	Detailed Description	32
4.3.2	Member Function Documentation	32
4.3.2.1	channelWithSources:	32
4.3.2.2	closeOSResources	33
4.3.2.3	initWithSources:	33
4.3.2.4	onFadeComplete:	33
4.3.2.5	onPanComplete:	33
4.3.2.6	onPitchComplete:	33
4.3.2.7	resetToDefault	33
4.3.3	Member Data Documentation	33
4.3.3.1	coneInnerAngle	33
4.3.3.2	coneOuterAngle	33
4.3.3.3	coneOuterGain	34
4.3.3.4	currentFadeCallbackCount	34
4.3.3.5	currentPanCallbackCount	34
4.3.3.6	currentPitchCallbackCount	34
4.3.3.7	direction	34
4.3.3.8	expectedFadeCallbackCount	34
4.3.3.9	expectedPanCallbackCount	34
4.3.3.10	expectedPitchCallbackCount	34
4.3.3.11	fadeCompleteSelector	34

4.3.3.12	<a href="#">fadeCompleteTarget</a>	34
4.3.3.13	<a href="#">gain</a>	35
4.3.3.14	<a href="#">interruptible</a>	35
4.3.3.15	<a href="#">looping</a>	35
4.3.3.16	<a href="#">maxDistance</a>	35
4.3.3.17	<a href="#">maxGain</a>	35
4.3.3.18	<a href="#">minGain</a>	35
4.3.3.19	<a href="#">muted</a>	35
4.3.3.20	<a href="#">panCompleteSelector</a>	35
4.3.3.21	<a href="#">panCompleteTarget</a>	36
4.3.3.22	<a href="#">paused</a>	36
4.3.3.23	<a href="#">pitch</a>	36
4.3.3.24	<a href="#">pitchCompleteSelector</a>	36
4.3.3.25	<a href="#">pitchCompleteTarget</a>	36
4.3.3.26	<a href="#">position</a>	36
4.3.3.27	<a href="#">referenceDistance</a>	36
4.3.3.28	<a href="#">rolloffFactor</a>	36
4.3.3.29	<a href="#">sourceRelative</a>	36
4.3.3.30	<a href="#">sourceType</a>	37
4.3.3.31	<a href="#">velocity</a>	37
4.3.4	<a href="#">Property Documentation</a>	37
4.3.4.1	<a href="#">context</a>	37
4.3.4.2	<a href="#">reservedSources</a>	37
4.3.4.3	<a href="#">sourcePool</a>	37
4.4	<a href="#">ALContext Class Reference</a>	37
4.4.1	<a href="#">Detailed Description</a>	40
4.4.2	<a href="#">Member Function Documentation</a>	40
4.4.2.1	<a href="#">clearBuffers</a>	40
4.4.2.2	<a href="#">close</a>	40
4.4.2.3	<a href="#">closeOSResources</a>	41
4.4.2.4	<a href="#">contextOnDevice:attributes:</a>	41
4.4.2.5	<a href="#">contextOnDevice:outputFrequency:refreshIntervals:synchronousContext:monoSources:stereoSources:</a>	41
4.4.2.6	<a href="#">ensureContextIsCurrent</a>	41
4.4.2.7	<a href="#">getProcAddress:</a>	42
4.4.2.8	<a href="#">initOnDevice:attributes:</a>	42
4.4.2.9	<a href="#">initOnDevice:outputFrequency:refreshIntervals:synchronousContext:monoSources:stereoSources:</a>	42
4.4.2.10	<a href="#">isExtensionPresent:</a>	43
4.4.2.11	<a href="#">notifySourceDeallocating:</a>	43
4.4.2.12	<a href="#">notifySourceInitializing:</a>	43
4.4.2.13	<a href="#">process</a>	43
4.4.2.14	<a href="#">setSuspended:</a>	44
4.4.2.15	<a href="#">stopAllSounds</a>	44
4.4.3	<a href="#">Member Data Documentation</a>	44
4.4.3.1	<a href="#">attributes</a>	44
4.4.3.2	<a href="#">sources</a>	44
4.4.3.3	<a href="#">suspended</a>	44
4.4.3.4	<a href="#">suspendHandler</a>	44
4.4.4	<a href="#">Property Documentation</a>	44
4.4.4.1	<a href="#">alVersion</a>	44
4.4.4.2	<a href="#">attributes</a>	44

4.4.4.3	context	45
4.4.4.4	device	45
4.4.4.5	distanceModel	45
4.4.4.6	dopplerFactor	45
4.4.4.7	extensions	45
4.4.4.8	listener	45
4.4.4.9	renderer	45
4.4.4.10	sources	45
4.4.4.11	speedOfSound	46
4.4.4.12	vendor	46
4.5	ALDevice Class Reference	46
4.5.1	Detailed Description	48
4.5.2	Member Function Documentation	48
4.5.2.1	clearBuffers	48
4.5.2.2	close	48
4.5.2.3	closeOSResources	48
4.5.2.4	deviceWithDeviceSpecifier:	48
4.5.2.5	getProcAddress:	48
4.5.2.6	initWithDeviceSpecifier:	49
4.5.2.7	isExtensionPresent:	49
4.5.2.8	notifyContextDeallocating:	49
4.5.2.9	notifyContextInitializing:	49
4.5.3	Member Data Documentation	49
4.5.3.1	contexts	49
4.5.3.2	suspendHandler	50
4.5.4	Property Documentation	50
4.5.4.1	contexts	50
4.5.4.2	device	50
4.5.4.3	extensions	50
4.5.4.4	majorVersion	50
4.5.4.5	minorVersion	50
4.6	ALListener Class Reference	50
4.6.1	Detailed Description	52
4.6.2	Member Function Documentation	52
4.6.2.1	initWithContext:	52
4.6.2.2	listenerForContext:	52
4.6.3	Member Data Documentation	52
4.6.3.1	suspendHandler	52
4.6.4	Property Documentation	53
4.6.4.1	context	53
4.6.4.2	gain	53
4.6.4.3	muted	53
4.6.4.4	orientation	53
4.6.4.5	position	53
4.6.4.6	velocity	53
4.7	ALOrientation Struct Reference	53
4.7.1	Detailed Description	54
4.7.2	Member Data Documentation	54
4.7.2.1	at	54
4.7.2.2	up	54

4.8	ALPoint Struct Reference	54
4.8.1	Detailed Description	55
4.8.2	Member Data Documentation	55
4.8.2.1	x	55
4.8.2.2	y	55
4.8.2.3	z	55
4.9	<ALSoundSource> Protocol Reference	55
4.9.1	Detailed Description	58
4.9.2	Member Function Documentation	58
4.9.2.1	clear	58
4.9.2.2	close	58
4.9.2.3	fadeTo:duration:target:selector:	58
4.9.2.4	panTo:duration:target:selector:	59
4.9.2.5	pitchTo:duration:target:selector:	59
4.9.2.6	play:	59
4.9.2.7	play:gain:pitch:pan:loop:	60
4.9.2.8	play:loop:	60
4.9.2.9	rewind	60
4.9.2.10	stop	60
4.9.2.11	stopActions	60
4.9.2.12	stopFade	60
4.9.2.13	stopPan	61
4.9.2.14	stopPitch	61
4.9.3	Property Documentation	61
4.9.3.1	coneInnerAngle	61
4.9.3.2	coneOuterAngle	61
4.9.3.3	coneOuterGain	61
4.9.3.4	direction	61
4.9.3.5	gain	61
4.9.3.6	interruptible	61
4.9.3.7	looping	62
4.9.3.8	maxDistance	62
4.9.3.9	maxGain	62
4.9.3.10	minGain	62
4.9.3.11	muted	62
4.9.3.12	pan	62
4.9.3.13	paused	62
4.9.3.14	pitch	62
4.9.3.15	playing	63
4.9.3.16	position	63
4.9.3.17	referenceDistance	63
4.9.3.18	rolloffFactor	63
4.9.3.19	sourceRelative	63
4.9.3.20	sourceType	63
4.9.3.21	velocity	63
4.9.3.22	volume	63
4.10	ALSoundSourcePool Class Reference	64
4.10.1	Detailed Description	65
4.10.2	Member Function Documentation	65
4.10.2.1	addSource:	65

4.10.2.2	close	65
4.10.2.3	closeOSResources	65
4.10.2.4	getFreeSource:	65
4.10.2.5	moveToHead:	65
4.10.2.6	pool	66
4.10.2.7	removeSource:	66
4.10.3	Member Data Documentation	66
4.10.3.1	sources	66
4.10.4	Property Documentation	66
4.10.4.1	sources	66
4.11	ALSource Class Reference	66
4.11.1	Detailed Description	69
4.11.2	Member Function Documentation	69
4.11.2.1	closeOSResources	69
4.11.2.2	delayedResumePlayback	69
4.11.2.3	initOnContext:	69
4.11.2.4	play	70
4.11.2.5	queueBuffer:	70
4.11.2.6	queueBuffers:	70
4.11.2.7	setSuspended:	70
4.11.2.8	source	70
4.11.2.9	sourceOnContext:	70
4.11.2.10	unqueueBuffer:	71
4.11.2.11	unqueueBuffers:	71
4.11.3	Member Data Documentation	71
4.11.3.1	abortPlaybackResume	71
4.11.3.2	gain	71
4.11.3.3	gainAction	71
4.11.3.4	interruptible	72
4.11.3.5	muted	72
4.11.3.6	panAction	72
4.11.3.7	pitchAction	72
4.11.3.8	shadowState	72
4.11.3.9	suspendHandler	72
4.11.4	Property Documentation	72
4.11.4.1	buffer	72
4.11.4.2	buffersProcessed	72
4.11.4.3	buffersQueued	72
4.11.4.4	context	73
4.11.4.5	offsetInBytes	73
4.11.4.6	offsetInSamples	73
4.11.4.7	offsetInSeconds	73
4.11.4.8	sourceId	73
4.11.4.9	state	73
4.12	ALVector Struct Reference	73
4.12.1	Detailed Description	74
4.12.2	Member Data Documentation	74
4.12.2.1	x	74
4.12.2.2	y	74
4.12.2.3	z	74



4.13 ALWrapper Class Reference	74
4.13.1 Detailed Description	82
4.13.2 Member Function Documentation	82
4.13.2.1 buffer3f:parameter:v1:v2:v3:	82
4.13.2.2 buffer3i:parameter:v1:v2:v3:	82
4.13.2.3 bufferData:format:data:size:frequency:	82
4.13.2.4 bufferDataStatic:format:data:size:frequency:	83
4.13.2.5 bufferf:parameter:value:	83
4.13.2.6 bufferfv:parameter:values:	83
4.13.2.7 bufferi:parameter:value:	84
4.13.2.8 bufferiv:parameter:values:	84
4.13.2.9 captureSamples:buffer:numSamples:	84
4.13.2.10 checkIfSuccessful	85
4.13.2.11 checkIfSuccessfulWithDevice	85
4.13.2.12 closeCaptureDevice:	85
4.13.2.13 closeDevice:	85
4.13.2.14 createContext:attributes:	86
4.13.2.15 decodeNullSeparatedStringList:	86
4.13.2.16 decodeSpaceSeparatedStringList:	86
4.13.2.17 deleteBuffer:	87
4.13.2.18 deleteBuffers:numBuffers:	87
4.13.2.19 deleteSource:	87
4.13.2.20 deleteSources:numSources:	87
4.13.2.21 destroyContext:	88
4.13.2.22 disable:	88
4.13.2.23 distanceModel:	88
4.13.2.24 dopplerFactor:	88
4.13.2.25 enable:	89
4.13.2.26 genBuffer	89
4.13.2.27 genBuffers:numBuffers:	89
4.13.2.28 genSource	89
4.13.2.29 genSources:numSources:	90
4.13.2.30 getBoolean:	90
4.13.2.31 getBooleanv:values:	90
4.13.2.32 getBuffer3f:parameter:v1:v2:v3:	90
4.13.2.33 getBuffer3i:parameter:v1:v2:v3:	91
4.13.2.34 getBufferf:parameter:	91
4.13.2.35 getBufferfv:parameter:values:	91
4.13.2.36 getBufferi:parameter:	92
4.13.2.37 getBufferiv:parameter:values:	92
4.13.2.38 getContextsDevice:	92
4.13.2.39 getContextsDevice:deviceReference:	93
4.13.2.40 getCurrentContext	93
4.13.2.41 getDouble:	93
4.13.2.42 getDoublev:values:	93
4.13.2.43 getEnumValue:	94
4.13.2.44 getEnumValue:name:	94
4.13.2.45 getFloat:	94
4.13.2.46 getFloatv:values:	94
4.13.2.47 getInteger:	95

4.13.2.48	<a href="#">getInteger:attribute:</a>	95
4.13.2.49	<a href="#">getInteger:attribute:size:data:</a>	95
4.13.2.50	<a href="#">getInteger:values:</a>	96
4.13.2.51	<a href="#">getListener3f:v1:v2:v3:</a>	96
4.13.2.52	<a href="#">getListener3i:v1:v2:v3:</a>	96
4.13.2.53	<a href="#">getListenerf:</a>	96
4.13.2.54	<a href="#">getListenerfv:values:</a>	97
4.13.2.55	<a href="#">getListeneri:</a>	97
4.13.2.56	<a href="#">getListeneriv:values:</a>	97
4.13.2.57	<a href="#">getMixerOutputDataRate</a>	98
4.13.2.58	<a href="#">getNullSeparatedStringList:</a>	98
4.13.2.59	<a href="#">getNullSeparatedStringList:attribute:</a>	98
4.13.2.60	<a href="#">getProcAddress:</a>	98
4.13.2.61	<a href="#">getProcAddress:name:</a>	99
4.13.2.62	<a href="#">getSource3f:parameter:v1:v2:v3:</a>	99
4.13.2.63	<a href="#">getSource3i:parameter:v1:v2:v3:</a>	99
4.13.2.64	<a href="#">getSourcef:parameter:</a>	100
4.13.2.65	<a href="#">getSourcefv:parameter:values:</a>	100
4.13.2.66	<a href="#">getSourcei:parameter:</a>	100
4.13.2.67	<a href="#">getSourceiv:parameter:values:</a>	101
4.13.2.68	<a href="#">getSpaceSeparatedStringList:</a>	101
4.13.2.69	<a href="#">getSpaceSeparatedStringList:attribute:</a>	101
4.13.2.70	<a href="#">getString:</a>	101
4.13.2.71	<a href="#">getString:attribute:</a>	102
4.13.2.72	<a href="#">isBuffer:</a>	102
4.13.2.73	<a href="#">isEnabled:</a>	102
4.13.2.74	<a href="#">isExtensionPresent:</a>	102
4.13.2.75	<a href="#">isExtensionPresent:name:</a>	103
4.13.2.76	<a href="#">isSource:</a>	103
4.13.2.77	<a href="#">listener3f:v1:v2:v3:</a>	103
4.13.2.78	<a href="#">listener3i:v1:v2:v3:</a>	104
4.13.2.79	<a href="#">listenerf:value:</a>	104
4.13.2.80	<a href="#">listenerfv:values:</a>	104
4.13.2.81	<a href="#">listeneri:value:</a>	105
4.13.2.82	<a href="#">listeneriv:values:</a>	105
4.13.2.83	<a href="#">makeContextCurrent:</a>	105
4.13.2.84	<a href="#">makeContextCurrent:deviceReference:</a>	105
4.13.2.85	<a href="#">openCaptureDevice:frequency:format:bufferSize:</a>	106
4.13.2.86	<a href="#">openDevice:</a>	106
4.13.2.87	<a href="#">processContext:</a>	106
4.13.2.88	<a href="#">setMixerOutputDataRate:</a>	106
4.13.2.89	<a href="#">source3f:parameter:v1:v2:v3:</a>	107
4.13.2.90	<a href="#">source3i:parameter:v1:v2:v3:</a>	107
4.13.2.91	<a href="#">sourcef:parameter:value:</a>	107
4.13.2.92	<a href="#">sourcefv:parameter:values:</a>	108
4.13.2.93	<a href="#">sourcei:parameter:value:</a>	108
4.13.2.94	<a href="#">sourceiv:parameter:values:</a>	108
4.13.2.95	<a href="#">sourcePause:</a>	109
4.13.2.96	<a href="#">sourcePausev:numSources:</a>	109
4.13.2.97	<a href="#">sourcePlay:</a>	109

4.13.2.98	sourcePlayv:numSources:	109
4.13.2.99	sourceQueueBuffers:numBuffers:bufferIds:	110
4.13.2.100	sourceRewind:	110
4.13.2.101	sourceRewindv:numSources:	110
4.13.2.102	sourceStop:	110
4.13.2.103	sourceStopv:numSources:	111
4.13.2.104	sourceUnqueueBuffers:numBuffers:bufferIds:	111
4.13.2.105	speedOfSound:	111
4.13.2.106	startCapture:	112
4.13.2.107	stopCapture:	112
4.13.2.108	suspendContext:	112
4.14	IOSVersion Class Reference	112
4.14.1	Detailed Description	113
4.14.2	Member Function Documentation	113
4.14.2.1	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	113
4.14.3	Property Documentation	113
4.14.3.1	version	113
4.15	NSMutableArray Class Reference	113
4.16	OAL_AsyncALBufferLoadOperation Class Reference	114
4.16.1	Detailed Description	114
4.16.2	Member Function Documentation	114
4.16.2.1	initWithUrl:reduceToMono:target:selector:	114
4.16.2.2	operationWithUrl:reduceToMono:target:selector:	115
4.16.3	Member Data Documentation	115
4.16.3.1	reduceToMono	115
4.16.3.2	selector	115
4.16.3.3	target	115
4.16.3.4	url	115
4.17	OAL_AsyncAudioTrackOperation Class Reference	116
4.17.1	Detailed Description	116
4.17.2	Member Function Documentation	117
4.17.2.1	initWithTrack:url:seekTime:target:selector:	117
4.17.2.2	operationWithTrack:url:seekTime:target:selector:	117
4.17.3	Member Data Documentation	117
4.17.3.1	audioTrack	117
4.17.3.2	seekTime	117
4.17.3.3	selector	117
4.17.3.4	target	117
4.17.3.5	url	118
4.18	OAL_AsyncAudioTrackPlayOperation Class Reference	118
4.18.1	Detailed Description	118
4.18.2	Member Function Documentation	119
4.18.2.1	initWithTrack:url:loops:target:selector:	119
4.18.2.2	operationWithTrack:url:loops:target:selector:	119
4.18.3	Member Data Documentation	119
4.18.3.1	loops	119
4.19	OAL_AsyncAudioTrackPreloadOperation Class Reference	120
4.19.1	Detailed Description	120
4.20	<OAL_GainProtocol> Protocol Reference	120
4.20.1	Detailed Description	120

4.20.2	Property Documentation	120
4.20.2.1	gain	120
4.21	<OAL_PanProtocol> Protocol Reference	121
4.21.1	Detailed Description	121
4.21.2	Property Documentation	121
4.21.2.1	pan	121
4.22	<OAL_PitchProtocol> Protocol Reference	121
4.22.1	Detailed Description	121
4.22.2	Property Documentation	121
4.22.2.1	pitch	121
4.23	<OAL_PositionProtocol> Protocol Reference	122
4.23.1	Detailed Description	122
4.23.2	Property Documentation	122
4.23.2.1	position	122
4.24	OALAction Class Reference	122
4.24.1	Detailed Description	124
4.24.2	Member Function Documentation	124
4.24.2.1	initWithDuration:	124
4.24.2.2	prepareWithTarget:	124
4.24.2.3	runWithTarget:	125
4.24.2.4	startAction	125
4.24.2.5	stopAction	125
4.24.2.6	updateCompletion:	125
4.24.3	Member Data Documentation	125
4.24.3.1	runningInManager	125
4.24.4	Property Documentation	125
4.24.4.1	duration	125
4.24.4.2	elapsed	125
4.24.4.3	running	125
4.24.4.4	target	126
4.25	OALActionManager Class Reference	126
4.25.1	Detailed Description	127
4.25.2	Member Function Documentation	127
4.25.2.1	doResetTimeDelta:	127
4.25.2.2	notifyActionStarted:	127
4.25.2.3	notifyActionStopped:	127
4.25.2.4	stopAllActions	127
4.25.2.5	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	128
4.25.3	Member Data Documentation	128
4.25.3.1	actionsToAdd	128
4.25.3.2	actionsToRemove	128
4.25.3.3	lastTimestamp	128
4.25.3.4	stepTimer	128
4.25.3.5	targetActions	128
4.25.3.6	targets	128
4.26	OALAudioFile Class Reference	128
4.26.1	Detailed Description	130
4.26.2	Member Function Documentation	130
4.26.2.1	audioDataWithStartFrame:numFrames:bufferSize:	130
4.26.2.2	bufferFromUrl:reduceToMono:	130

4.26.2.3	<a href="#">bufferNamed:startFrame:numFrames:</a>	131
4.26.2.4	<a href="#">close</a>	131
4.26.2.5	<a href="#">closeOSResources</a>	131
4.26.2.6	<a href="#">fileWithUrl:reduceToMono:</a>	131
4.26.2.7	<a href="#">initWithUrl:reduceToMono:</a>	131
4.26.3	<a href="#">Member Data Documentation</a>	132
4.26.3.1	<a href="#">fileHandle</a>	132
4.26.3.2	<a href="#">originalChannelsPerFrame</a>	132
4.26.3.3	<a href="#">streamDescription</a>	132
4.26.4	<a href="#">Property Documentation</a>	132
4.26.4.1	<a href="#">reduceToMono</a>	132
4.26.4.2	<a href="#">streamDescription</a>	132
4.26.4.3	<a href="#">totalFrames</a>	132
4.26.4.4	<a href="#">url</a>	132
4.27	<a href="#">OALAudioSession Class Reference</a>	133
4.27.1	<a href="#">Detailed Description</a>	135
4.27.2	<a href="#">Member Function Documentation</a>	135
4.27.2.1	<a href="#">close</a>	135
4.27.2.2	<a href="#">closeOSResources</a>	136
4.27.2.3	<a href="#">forceEndInterruption</a>	136
4.27.2.4	<a href="#">getFloatProperty:</a>	136
4.27.2.5	<a href="#">getIntProperty:</a>	136
4.27.2.6	<a href="#">getStringProperty:</a>	136
4.27.2.7	<a href="#">onAudioError:</a>	137
4.27.2.8	<a href="#">setAudioMode</a>	137
4.27.2.9	<a href="#">setIntProperty:value:</a>	137
4.27.2.10	<a href="#">setSuspended:</a>	137
4.27.2.11	<a href="#">SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER</a>	137
4.27.2.12	<a href="#">updateFromAudioSessionCategory</a>	137
4.27.2.13	<a href="#">updateFromFlags</a>	137
4.27.3	<a href="#">Member Data Documentation</a>	138
4.27.3.1	<a href="#">audioSessionWasActive</a>	138
4.27.3.2	<a href="#">lastResetTime</a>	138
4.27.3.3	<a href="#">suspendHandler</a>	138
4.27.4	<a href="#">Property Documentation</a>	138
4.27.4.1	<a href="#">allowIpod</a>	138
4.27.4.2	<a href="#">audioRoute</a>	138
4.27.4.3	<a href="#">audioSessionActive</a>	138
4.27.4.4	<a href="#">audioSessionCategory</a>	138
4.27.4.5	<a href="#">audioSessionDelegate</a>	139
4.27.4.6	<a href="#">handleInterruptions</a>	139
4.27.4.7	<a href="#">hardwareMuted</a>	139
4.27.4.8	<a href="#">hardwareVolume</a>	139
4.27.4.9	<a href="#">honorSilentSwitch</a>	139
4.27.4.10	<a href="#">ipodDucking</a>	139
4.27.4.11	<a href="#">ipodPlaying</a>	140
4.27.4.12	<a href="#">useHardwareIfAvailable</a>	140
4.28	<a href="#">OALAudioTrack Class Reference</a>	140
4.28.1	<a href="#">Detailed Description</a>	144
4.28.2	<a href="#">Member Function Documentation</a>	145

4.28.2.1	averagePowerForChannel:	145
4.28.2.2	clear	145
4.28.2.3	close	145
4.28.2.4	closeOSResources	145
4.28.2.5	fadeTo:duration:target:selector:	145
4.28.2.6	panTo:duration:target:selector:	146
4.28.2.7	peakPowerForChannel:	146
4.28.2.8	play	146
4.28.2.9	playAtTime:	146
4.28.2.10	playFile:	147
4.28.2.11	playFile:loops:	147
4.28.2.12	playFileAsync:loops:target:selector:	147
4.28.2.13	playFileAsync:target:selector:	147
4.28.2.14	playUrl:	148
4.28.2.15	playUrl:loops:	148
4.28.2.16	playUrlAsync:loops:target:selector:	148
4.28.2.17	playUrlAsync:target:selector:	148
4.28.2.18	preloadFile:	149
4.28.2.19	preloadFile:seekTime:	149
4.28.2.20	preloadFileAsync:seekTime:target:selector:	149
4.28.2.21	preloadFileAsync:target:selector:	150
4.28.2.22	preloadUrl:	150
4.28.2.23	preloadUrl:seekTime:	150
4.28.2.24	preloadUrlAsync:seekTime:target:selector:	150
4.28.2.25	preloadUrlAsync:target:selector:	151
4.28.2.26	setSuspended:	151
4.28.2.27	stop	151
4.28.2.28	stopActions	151
4.28.2.29	stopFade	151
4.28.2.30	stopPan	152
4.28.2.31	track	152
4.28.2.32	updateMeters	152
4.28.3	Member Data Documentation	152
4.28.3.1	gainAction	152
4.28.3.2	interrupted	152
4.28.3.3	operationQueue	152
4.28.3.4	panAction	152
4.28.3.5	simulatorPlayerRef	153
4.28.3.6	suspendHandler	153
4.28.4	Property Documentation	153
4.28.4.1	autoPreload	153
4.28.4.2	currentlyLoadedUrl	153
4.28.4.3	currentTime	153
4.28.4.4	delegate	153
4.28.4.5	deviceCurrentTime	153
4.28.4.6	duration	154
4.28.4.7	gain	154
4.28.4.8	meteringEnabled	154
4.28.4.9	muted	154
4.28.4.10	numberOfChannels	154

4.28.4.11	numberOfLoops	154
4.28.4.12	pan	154
4.28.4.13	paused	154
4.28.4.14	player	155
4.28.4.15	playing	155
4.28.4.16	preloaded	155
4.28.4.17	volume	155
4.29	OALAudioTracks Class Reference	155
4.29.1	Detailed Description	157
4.29.2	Member Function Documentation	157
4.29.2.1	close	157
4.29.2.2	closeOSResources	157
4.29.2.3	notifyTrackDeallocating:	157
4.29.2.4	notifyTrackInitializing:	157
4.29.2.5	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	157
4.29.3	Member Data Documentation	157
4.29.3.1	suspendHandler	157
4.29.3.2	tracks	157
4.29.4	Property Documentation	158
4.29.4.1	muted	158
4.29.4.2	paused	158
4.29.4.3	tracks	158
4.30	OALCallAction Class Reference	158
4.30.1	Detailed Description	159
4.30.2	Member Function Documentation	159
4.30.2.1	actionWithCallTarget:selector:	159
4.30.2.2	actionWithCallTarget:selector:withObject:	160
4.30.2.3	actionWithCallTarget:selector:withObject:withObject:	160
4.30.2.4	initWithCallTarget:selector:	160
4.30.2.5	initWithCallTarget:selector:withObject:	161
4.30.2.6	initWithCallTarget:selector:withObject:withObject:	161
4.30.3	Member Data Documentation	161
4.30.3.1	callTarget	161
4.30.3.2	numObjects	161
4.30.3.3	object1	161
4.30.3.4	object2	162
4.30.3.5	selector	162
4.31	OALConcurrentActions Class Reference	162
4.31.1	Detailed Description	163
4.31.2	Member Function Documentation	163
4.31.2.1	actions:	163
4.31.2.2	actionsFromArray:	163
4.31.2.3	initWithActions:	164
4.31.3	Member Data Documentation	164
4.31.3.1	actionsWithDuration	164
4.31.3.2	pDurations	164
4.31.4	Property Documentation	164
4.31.4.1	actions	164
4.32	OALExponentialFunction Class Reference	164
4.32.1	Detailed Description	165

4.32.2	Member Function Documentation	165
4.32.2.1	function	165
4.32.2.2	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	166
4.33	<OALFunction> Protocol Reference	166
4.33.1	Detailed Description	166
4.33.2	Member Function Documentation	166
4.33.2.1	valueForInput:	166
4.34	OALFunctionAction Class Reference	167
4.34.1	Detailed Description	168
4.34.2	Member Function Documentation	168
4.34.2.1	actionWithDuration:endValue:	168
4.34.2.2	actionWithDuration:endValue:function:	169
4.34.2.3	actionWithDuration:startValue:endValue:function:	169
4.34.2.4	defaultFunction	169
4.34.2.5	initWithDuration:endValue:	170
4.34.2.6	initWithDuration:endValue:function:	170
4.34.2.7	initWithDuration:startValue:endValue:function:	170
4.34.3	Member Data Documentation	171
4.34.3.1	delta	171
4.34.3.2	lowValue	171
4.34.3.3	realFunction	171
4.34.3.4	reverseFunction	171
4.34.4	Property Documentation	171
4.34.4.1	endValue	171
4.34.4.2	function	171
4.34.4.3	startValue	171
4.35	OALGainAction Class Reference	171
4.35.1	Detailed Description	172
4.36	OALLinearFunction Class Reference	172
4.36.1	Detailed Description	173
4.36.2	Member Function Documentation	173
4.36.2.1	function	173
4.36.2.2	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	173
4.37	OALLogarithmicFunction Class Reference	173
4.37.1	Detailed Description	174
4.37.2	Member Function Documentation	174
4.37.2.1	function	174
4.37.2.2	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	175
4.38	OALMoveByAction Class Reference	175
4.38.1	Detailed Description	176
4.38.2	Member Function Documentation	176
4.38.2.1	actionWithDuration:delta:	176
4.38.2.2	actionWithDuration:delta:unitsPerSecond:	176
4.38.2.3	initWithDuration:delta:	177
4.38.2.4	initWithDuration:delta:unitsPerSecond:	177
4.38.3	Member Data Documentation	177
4.38.3.1	startPoint	177
4.38.4	Property Documentation	177
4.38.4.1	delta	177
4.38.4.2	unitsPerSecond	177



4.39	OALMoveToAction Class Reference	178
4.39.1	Detailed Description	179
4.39.2	Member Function Documentation	179
4.39.2.1	actionWithDuration:position:	179
4.39.2.2	actionWithUnitsPerSecond:position:	179
4.39.2.3	initWithDuration:position:	179
4.39.2.4	initWithUnitsPerSecond:position:	180
4.39.3	Member Data Documentation	180
4.39.3.1	delta	180
4.39.3.2	startPoint	180
4.39.4	Property Documentation	180
4.39.4.1	position	180
4.39.4.2	unitsPerSecond	180
4.40	OALPanAction Class Reference	181
4.40.1	Detailed Description	181
4.41	OALPitchAction Class Reference	181
4.41.1	Detailed Description	181
4.42	OALPlaceAction Class Reference	182
4.42.1	Detailed Description	182
4.42.2	Member Function Documentation	183
4.42.2.1	actionWithPosition:	183
4.42.2.2	initWithPosition:	183
4.42.3	Property Documentation	183
4.42.3.1	position	183
4.43	OALReverseFunction Class Reference	183
4.43.1	Detailed Description	184
4.43.2	Member Function Documentation	184
4.43.2.1	functionWithFunction:	184
4.43.2.2	initWithFunction:	185
4.43.3	Property Documentation	185
4.43.3.1	function	185
4.44	OALSCurveFunction Class Reference	185
4.44.1	Detailed Description	186
4.44.2	Member Function Documentation	186
4.44.2.1	function	186
4.44.2.2	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	186
4.45	OALSequentialActions Class Reference	187
4.45.1	Detailed Description	188
4.45.2	Member Function Documentation	188
4.45.2.1	actions:	188
4.45.2.2	actionsFromArray:	188
4.45.2.3	initWithActions:	189
4.45.3	Member Data Documentation	189
4.45.3.1	actionIndex	189
4.45.3.2	currentAction	189
4.45.3.3	pCurrentActionComplete	189
4.45.3.4	pCurrentActionDuration	189
4.45.3.5	pDurations	189
4.45.3.6	pLastComplete	189
4.45.4	Property Documentation	189

4.45.4.1	actions	189
4.46	OALSimpleAudio Class Reference	190
4.46.1	Detailed Description	194
4.46.2	Member Function Documentation	194
4.46.2.1	close	194
4.46.2.2	closeOSResources	194
4.46.2.3	initWithSources:	194
4.46.2.4	internalPreloadEffect:reduceToMono:	194
4.46.2.5	playBg	195
4.46.2.6	playBg:	195
4.46.2.7	playBg:loop:	195
4.46.2.8	playBg:volume:pan:loop:	196
4.46.2.9	playBgWithLoop:	196
4.46.2.10	playBuffer:volume:pitch:pan:loop:	196
4.46.2.11	playEffect:	197
4.46.2.12	playEffect:loop:	197
4.46.2.13	playEffect:volume:pitch:pan:loop:	197
4.46.2.14	preloadBg:	198
4.46.2.15	preloadBg:seekTime:	198
4.46.2.16	preloadEffect:	198
4.46.2.17	preloadEffect:reduceToMono:	199
4.46.2.18	resetToDefault	199
4.46.2.19	sharedInstanceWithSources:	199
4.46.2.20	stopAllEffects	199
4.46.2.21	stopBg	199
4.46.2.22	stopEverything	200
4.46.2.23	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	200
4.46.2.24	unloadAllEffects	200
4.46.2.25	unloadEffect:	200
4.46.3	Member Data Documentation	200
4.46.3.1	channel	200
4.46.3.2	context	200
4.46.3.3	device	200
4.46.3.4	pendingLoadCount	200
4.46.3.5	preloadCache	201
4.46.4	Property Documentation	201
4.46.4.1	allowIpod	201
4.46.4.2	backgroundTrack	201
4.46.4.3	backgroundTrackURL	201
4.46.4.4	bgMuted	201
4.46.4.5	bgPaused	201
4.46.4.6	bgPlaying	201
4.46.4.7	bgVolume	201
4.46.4.8	effectsMuted	202
4.46.4.9	effectsPaused	202
4.46.4.10	effectsVolume	202
4.46.4.11	honorSilentSwitch	202
4.46.4.12	interrupted	202
4.46.4.13	manuallySuspended	202
4.46.4.14	muted	202

4.46.4.15	paused	202
4.46.4.16	preloadCacheCount	202
4.46.4.17	preloadCacheEnabled	202
4.46.4.18	reservedSources	203
4.46.4.19	suspended	203
4.46.4.20	useHardwareIfAvailable	203
4.47	OALSuspendHandler Class Reference	203
4.47.1	Detailed Description	205
4.47.2	Member Function Documentation	205
4.47.2.1	addSuspendListener:	205
4.47.2.2	handlerWithTarget:selector:	205
4.47.2.3	initWithTarget:selector:	206
4.47.2.4	removeSuspendListener:	206
4.47.3	Member Data Documentation	206
4.47.3.1	interruptLock	206
4.47.3.2	listeners	206
4.47.3.3	manualSuspendLock	206
4.47.3.4	manualSuspendStates	206
4.47.3.5	suspendStatusChangeSelector	206
4.47.3.6	suspendStatusChangeTarget	207
4.47.4	Property Documentation	207
4.47.4.1	interrupted	207
4.47.4.2	manuallySuspended	207
4.47.4.3	suspended	207
4.48	<OALSuspendListener> Protocol Reference	207
4.48.1	Detailed Description	208
4.48.2	Property Documentation	209
4.48.2.1	interrupted	209
4.48.2.2	manuallySuspended	209
4.49	<OALSuspendManager> Protocol Reference	209
4.49.1	Detailed Description	210
4.49.2	Member Function Documentation	211
4.49.2.1	addSuspendListener:	211
4.49.2.2	removeSuspendListener:	211
4.49.3	Property Documentation	211
4.49.3.1	suspended	211
4.50	OALTargetedAction Class Reference	211
4.50.1	Detailed Description	212
4.50.2	Member Function Documentation	212
4.50.2.1	actionWithTarget:action:	212
4.50.2.2	initWithTarget:action:	213
4.50.3	Member Data Documentation	213
4.50.3.1	action	213
4.50.4	Property Documentation	213
4.50.4.1	forcedTarget	213
4.51	OALTools Class Reference	213
4.51.1	Detailed Description	214
4.51.2	Member Function Documentation	214
4.51.2.1	notifyAudioSessionError:function:description:	214
4.51.2.2	notifyExtAudioError:function:description:	214

4.51.2.3	urlForPath:	214
4.52	OpenALManager Class Reference	215
4.52.1	Detailed Description	217
4.52.2	Member Function Documentation	217
4.52.2.1	bufferAsyncFromFile:reduceToMono:target:selector:	217
4.52.2.2	bufferAsyncFromFile:target:selector:	218
4.52.2.3	bufferAsyncFromUrl:reduceToMono:target:selector:	218
4.52.2.4	bufferAsyncFromUrl:target:selector:	219
4.52.2.5	bufferFromFile:	219
4.52.2.6	bufferFromFile:reduceToMono:	220
4.52.2.7	bufferFromUrl:	220
4.52.2.8	bufferFromUrl:reduceToMono:	220
4.52.2.9	clearAllBuffers	221
4.52.2.10	close	221
4.52.2.11	closeOSResources	221
4.52.2.12	notifyDeviceDeallocating:	221
4.52.2.13	notifyDeviceInitializing:	221
4.52.2.14	setSuspended:	221
4.52.2.15	SYNTHESIZE_SINGLETON_FOR_CLASS_HEADER	221
4.52.3	Member Data Documentation	221
4.52.3.1	devices	221
4.52.3.2	operationQueue	222
4.52.3.3	suspendHandler	222
4.52.4	Property Documentation	222
4.52.4.1	availableCaptureDevices	222
4.52.4.2	availableDevices	222
4.52.4.3	currentContext	222
4.52.4.4	defaultCaptureDeviceSpecifier	222
4.52.4.5	defaultDeviceSpecifier	222
4.52.4.6	devices	222
4.52.4.7	mixerOutputFrequency	222

# Chapter 1

## ObjectAL for iPhone

iOS Audio development, minus the headache.

Version 2.0

Copyright 2009-2011 Karl Stenerud

Released under the [Apache License v2.0](#)

### 1.1 Contents

- [Introduction](#)
- [ObjectAL and OpenAL](#)
- [Adding ObjectAL to your project](#) (also, installing the documentation into XCode)
- [Compile-Time Configuration](#)
- [Audio Formats](#)
- [Choosing Playback Types](#)
- [Using OALSimpleAudio](#)
- [Using the OpenAL Objects and OALAudioTrack](#)
- [Other Examples](#)
- [iOS Issues that can impede playback](#)
- [Simulator Issues](#)

### 1.2 Introduction

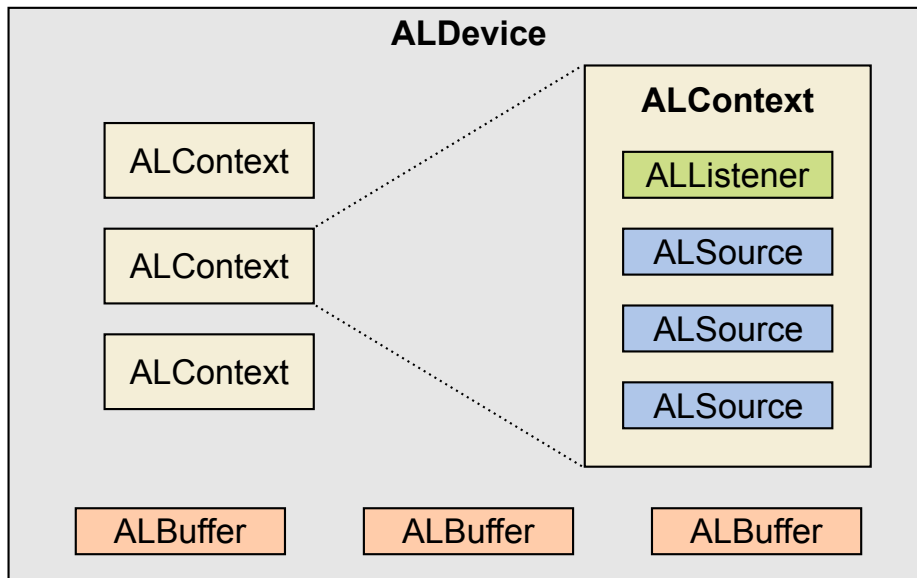
**ObjectAL for iPhone** is designed to be a simpler, more intuitive interface to OpenAL and AVAudioPlayer. There are four main parts to **ObjectAL for iPhone**:

OALSimpleAudio (Simpler Interface)			
ObjectAL (Sound Effects)		OALAudioSession (Session Management)	OALAudioTrack (Long-play Audio)
OpenAL	ExtAudio	AudioSession	AVAudioPlayer

- [ObjectAL](#) gives you full access to the OpenAL system without the hassle of the C API. All OpenAL operations can be performed using first class objects and properties, without needing to muddle around with arrays of data, maintain IDs, or pass around pointers to basic types. [ObjectALManager](#) also provides sound loading routines.
- [OALAudioTrack](#) provides a simpler interface to [AVAudioPlayer](#), allowing you to play, stop, pause, fade, and mute background music tracks.
- [OALAudioSession](#) handles audio session management in iOS devices, and provides an easy way to configure session behavior such as how to handle iPod-style music and the silent switch.
- [OALSimpleAudio](#) layers on top of the other three, providing an even simpler interface for playing background music and sound effects.

### 1.3 ObjectAL and OpenAL

[ObjectAL](#) follows the same basic principles as the [OpenAL API by Creative Labs](#).



- **OpenALManager** provides some overall controls that affect everything, manages the current context, and provides audio loading routines.
- **ALDevice** represents a physical audio device.  
Each device can have one or more contexts (**ALContext**) created on it, and can have multiple buffers (**ALBuffer**) associated with it.
- **ALContext** controls the overall sound environment, such as distance model, doppler effect, and speed of sound.  
Each context has one listener (**ALListener**), and can have multiple sources (**ALSource**) opened on it (up to a maximum of 32 overall on iPhone).
- **ALListener** represents the listener of sounds originating on its context (one listener per context). It has position, orientation, and velocity.
- **ALSource** is a sound emitting source that plays sound data from an **ALBuffer**. It has position, direction, velocity, as well as other properties which determine how the sound is emitted.
- **ALChannelSource** allows you to reserve a certain number of sources for special purposes.
- **ALBuffer** is simply a container for sound data. Only linear PCM is supported directly, but **OpenALManager** load methods, and **OALSimpleAudio** effect preload and play methods, will automatically convert any formats that don't require hardware decoding (though conversion results in a longer loading time).

**Note:** While OpenAL allows for multiple devices and contexts, in practice you'll only use one device and one context when using OpenAL under iOS.

Further information regarding the more advanced features of OpenAL (such as distance models) are available via the [OpenAL Documentation at Creative Labs](#).

In particular, read up on the various property values for sources and listeners (such as Doppler Shift) in the [OpenAL Programmer's Guide](#), and distance models in section 3 of the [OpenAL Specification](#).

## 1.4 Adding ObjectAL to your project

To add ObjectAL to your project, do the following:

1. Copy libs/ObjectAL from this project into your project. You can simply drag it into the "Groups & Files" section in xcode if you like (be sure to select "Copy items into destination group's folder").

Alternatively, you can build ObjectAL as a static library (as it's configured to do in the ObjectAL demo project).

2. Add the following frameworks to your project:

- OpenAL.framework
- AudioToolbox.framework
- AVFoundation.framework

3. Start using ObjectAL!

**Note:** The demos in this project use [Cocos2d](#), a very nice 2d game engine. However, ObjectAL doesn't require it. You can just as easily use ObjectAL in your Cocoa app or anything you wish.

**Note #2:** You do NOT have to provide a link to the Apache license from within your application. Simply including a copy of the license in your project is sufficient.

### 1.4.1 Installing the ObjectAL Documentation into XCode

By installing the ObjectAL documentation into XCode's Developer Documentation system, you gain the ability to look up ObjectAL classes and methods just like you'd look up Apple classes and methods. You can install the ObjectAL documentation into XCode's Developer Documentation system by doing the following:

1. Install [Doxygen](#). You can either use the OSX installer or MacPorts.
2. Build the "Documentation" target in this project.
3. Open the developer documentation and type "ObjectAL" into the search box.



## 1.5 Compile-Time Configuration

[ObjectALConfig.h](#) contains configuration defines that will affect at a high level how ObjectAL behaves. Look inside [ObjectALConfig.h](#) to see what can be configured, and what each configuration value does.

The recommended values are fine for most users, but Cocos2D users may want to set `OBJECTAL_USE_COCOS2D_ACTIONS` so that the audio actions (such as fade) use the Cocos2D action manager.

## 1.6 Audio Formats

The audio formats officially supported by Apple are [defined here](#).

### 1.6.1 OALAudioTrack Supported Formats

[OALAudioTrack](#) supports all hardware and software decoded formats.

### 1.6.2 OpenAL Supported Formats

OpenAL officially supports 8 or 16 bit PCM data only. However, Apple's implementation only seems to work with 16 bit data.

The effects preloading/playing methods in [OALSimpleAudio](#) and the buffer loading methods in [OpenALManager](#) can load any audio file that can be software decoded. However, there is a cost incurred at load time converting to a native OpenAL format. To avoid this, convert all of your samples to a CAFF container with 16-bit little endian integer PCM format and the same sample rate as "mixerOutputFrequency" in [OpenALManager](#) (by default, 44100Hz). Note, however, that uncompressed files can get quite large.

Convert to iOS native uncompressed format using Apple's "afconvert" command line tool:

```
afconvert -f caff -d LEI16@44100 sourcefile.wav destfile.caf
```

Alternatively, if sound file load time is not an issue for you, you can lower your app footprint size (for over-the-air app download) by using a compressed format.

Convert to AAC compressed format with CAFF container using Apple's "afconvert" command line tool:

```
afconvert -f caff -d aac sourcefile.wav destfile.caf
```

## 1.7 Choosing Playback Types

OpenAL ([ALSource](#), or effects in [OALSimpleAudio](#)) and **AVAudioPlayer** ([OALAudioTrack](#), or background audio in [OALSimpleAudio](#)) are playback technologies built for different

purposes. OpenAL is designed for game-style short sound effects that have no playback delay. AVAudioPlayer is designed for music playback. You can of course mix and match as you please.

	OpenAL	AVAudioPlayer
<b>Playback Delay</b>	None	Small delay if not preloaded
<b>Format on Disk</b>	Any software decodable format	Any software decodable format, or any hardware format if using hardware
<b>Decoding</b>	During load	During playback
<b>Memory Use</b>	Entire file loaded and decompressed into memory	File streamed realtime (very low memory use)
<b>Max Simult. Sources</b>	32	As many as the CPU can handle
<b>Playback Performance</b>	Good	Excellent with 1 track (if using hardware). Good with 2 tracks. Not so good with more (each non-hardware track taxes the CPU significantly, especially if the files are compressed).
<b>Looped Playback</b>	Yes (on or off)	Yes (specify number of loops or -1 = forever)
<b>Panning</b>	Yes (mono files only)	Yes (iOS 4.0+ only)
<b>Positional Audio</b>	Yes (mono files only)	No
<b>Modify Pitch</b>	Yes	No
<b>Audio Power Metering</b>	No	Yes

## 1.8 Using OALSimpleAudio

By far, the easiest component to use is [OALSimpleAudio](#). You sacrifice some power for ease-of-use, but for many projects it is more than sufficient. You can also use your own instances of [OALAudioTrack](#), [ALSource](#), [ALBuffer](#) and such alongside of [OALSimpleAudio](#) if you want (just be sure to set OALSimpleAudio's reservedSources to less than 32 if you want to make your own instances of [ALSource](#)).

Here is a code example using purely [OALSimpleAudio](#):

```
// OALSimpleAudioSample.h

@interface OALSimpleAudioSample : NSObject
{
    // No objects to keep track of...
}

@end
```

```
// OALSimpleAudioSample.m

#import "OALSimpleAudioSample.h"
#import "ObjectAL.h"

#define SHOOT_SOUND @"shoot.caf"
#define EXPLODE_SOUND @"explode.caf"

#define INGAME_MUSIC_FILE @"bg_music.mp3"
#define GAMEOVER_MUSIC_FILE @"gameover_music.mp3"

@implementation OALSimpleAudioSample

- (id) init
{
    if(nil != (self = [super init]))
    {
        // We don't want ipod music to keep playing since
        // we have our own bg music.
        [OALSimpleAudio sharedInstance].allowIpod = NO;

        // Mute all audio if the silent switch is turned on.
        [OALSimpleAudio sharedInstance].honorSilentSwitch = YES;

        // This loads the sound effects into memory so that
        // there's no delay when we tell it to play them.
        [[OALSimpleAudio sharedInstance] preloadEffect:SHOOT_SOUND];
        [[OALSimpleAudio sharedInstance] preloadEffect:EXPLODE_SOUND];
    }
    return self;
}

- (void) onGameStart
{
    // Play the BG music and loop it.
    [[OALSimpleAudio sharedInstance] playBg:INGAME_MUSIC_FILE loop:YES];
}

- (void) onGamePause
{
    [OALSimpleAudio sharedInstance].paused = YES;
}

- (void) onGameResume
{
    [OALSimpleAudio sharedInstance].paused = NO;
}

- (void) onGameOver
{
    // Could use stopEverything here if you want
    [[OALSimpleAudio sharedInstance] stopAllEffects];

    // We only play the game over music through once.
    [[OALSimpleAudio sharedInstance] playBg:GAMEOVER_MUSIC_FILE];
}

- (void) onShipShotABullet
```

```

{
    [[OALSimpleAudio sharedInstance] playEffect:SHOOT_SOUND];
}

- (void) onShipGotHit
{
    [[OALSimpleAudio sharedInstance] playEffect:EXPLODE_SOUND];
}

- (void) onQuitToMainMenu
{
    // Stop all music and sound effects.
    [[OALSimpleAudio sharedInstance] stopEverything];

    // Unload all sound effects and bg music so that it doesn't fill
    // memory unnecessarily.
    [[OALSimpleAudio sharedInstance] unloadAllEffects];
}

@end

```

## 1.9 Using the OpenAL Objects and OALAudioTrack

The OpenAL objects and [OALAudioTrack](#) offer you much more power at the cost of complexity. Here's the same thing as above, done using OpenAL components and [OALAudioTrack](#):

```

// OpenALAudioTrackSample.h

#import <Foundation/Foundation.h>
#import "ObjectAL.h"

@interface OpenALAudioTrackSample : NSObject
{
    // Sound Effects
    ALDevice* device;
    ALContext* context;
    ALChannelSource* channel;
    ALBuffer* shootBuffer;
    ALBuffer* explosionBuffer;

    // Background Music
    OALAudioTrack* musicTrack;
}

@end

// OpenALAudioTrackSample.m

#import "OpenALAudioTrackSample.h"

#define SHOOT_SOUND @"shoot.caf"
#define EXPLODE_SOUND @"explode.caf"

#define INGAME_MUSIC_FILE @"bg_music.mp3"

```

```
#define GAMEOVER_MUSIC_FILE @"gameover_music.mp3"

@implementation OpenALAudioTrackSample

- (id) init
{
    if(nil != (self = [super init]))
    {
        // Create the device and context.
        // Note that it's easier to just let OALSimpleAudio handle
        // these rather than make and manage them yourself.
        device = [[ALDevice deviceWithDeviceSpecifier:nil] retain];
        context = [[ALContext contextOnDevice:device attributes:nil] retain];
        [OpenALManager sharedInstance].currentContext = context;

        // Deal with interruptions for me!
        [OALAudioSession sharedInstance].handleInterruptions = YES;

        // We don't want ipod music to keep playing since
        // we have our own bg music.
        [OALAudioSession sharedInstance].allowIpod = NO;

        // Mute all audio if the silent switch is turned on.
        [OALAudioSession sharedInstance].honorSilentSwitch = YES;

        // Take all 32 sources for this channel.
        // (we probably won't use that many but what the heck!)
        channel = [[ALChannelSource channelWithSources:32] retain];

        // Preload the buffers so we don't have to load and play them later.
        shootBuffer = [[[OpenALManager sharedInstance]
                        bufferFromFile:SHOOT_SOUND] retain];
        explosionBuffer = [[[OpenALManager sharedInstance]
                           bufferFromFile:EXPLODE_SOUND] retain];

        // Background music track.
        musicTrack = [[OALAudioTrack track] retain];
    }
    return self;
}

- (void) dealloc
{
    [musicTrack release];

    [channel release];
    [shootBuffer release];
    [explosionBuffer release];

    // Note: You'll likely only have one device and context open throughout
    // your program, so in a real program you'd be better off making a
    // singleton object that manages the device and context, rather than
    // allocating/deallocating it here.
    // Most of the demos just let OALSimpleAudio manage the device and context
    // for them.
    [context release];
    [device release];

    [super dealloc];
}
```

```
- (void) onGameStart
{
    // Play the BG music and loop it forever.
    [musicTrack playFile:INGAME_MUSIC_FILE loops:-1];
}

- (void) onGamePause
{
    musicTrack.paused = YES;
    channel.paused = YES;
}

- (void) onGameResume
{
    channel.paused = NO;
    musicTrack.paused = NO;
}

- (void) onGameOver
{
    [channel stop];
    [musicTrack stop];

    // We only play the game over music through once.
    [musicTrack playFile:GAMEOVER_MUSIC_FILE];
}

- (void) onShipShotABullet
{
    [channel play:shootBuffer];
}

- (void) onShipGotHit
{
    [channel play:explosionBuffer];
}

- (void) onQuitToMainMenu
{
    // Stop all music and sound effects.
    [channel stop];
    [musicTrack stop];
}

@end
```

## 1.10 Other Examples

The demo scenes in this distribution have been crafted to demonstrate common uses of this library. Try them out and go through the code to see how it's done. I've done my best to keep the code readable. Really!

The current demos are:

- **SingleSourceDemo**: Demonstrates using a location based source and a listener.
- **TwoSourceDemo**: Demonstrates using two location based sources and a listener.
- **VolumePitchPanDemo**: Demonstrates using gain, pitch, and pan controls.

- **CrossFadeDemo**: Demonstrates crossfading between two sources.
- **ChannelsDemo**: Demonstrates using audio channels.
- **FadeDemo**: Demonstrates realtime fading with [OALAudioTrack](#) and [ALSource](#).
- **AudioTrackDemo**: Demonstrates using multiple [OALAudioTrack](#) objects.
- **HardwareDemo**: Demonstrates hardware monitoring features.
- **AudioSessionDemo**: Allows you to play with various audio session settings.
- **PlanetKillerDemo**: Demonstrates using [OALSimpleAudio](#) in a game setting.

## 1.11 iOS Issues that can impede playback

Certain versions of iOS have bugs or quirks, requiring workarounds. ObjectAL tries to handle most of these automatically, but there are cases that require specific handling by the developer. These are:

### 1.11.1 MPMoviePlayerController on iOS 3.x

In iOS 3.x, MPMoviePlayerController doesn't play nice, and takes over the audio session when you play a video. In order to mitigate this, you must manually suspend OpenAL, play the video, and then manually unsuspend once video playback finishes:

```
- (void) playVideo
{
    if([myMoviePlayer respondsToSelector:@selector(view)])
    {
        [myMoviePlayer setFullscreen:YES animated:YES];
    }
    else
    {
        // No "view" method means we are < 4.0
        // Manually suspend so iOS 3.x doesn't clobber our session!
        [OpenALManager sharedInstance].manuallySuspended = YES;
    }

    [myMoviePlayer play];

    [[NSNotificationCenter defaultCenter]
     addObserver:self
     selector:@selector(movieFinishedCallback:)
     name:MPMoviePlayerPlaybackDidFinishNotification
     object:myMoviePlayer];
}

-(void)movieFinishedCallback:(NSNotification *)notification
{
    if([myMoviePlayer respondsToSelector:@selector(view)])
    {
        if (myMoviePlayer.fullscreen)
        {
            [myMoviePlayer setFullscreen:NO animated:YES];
        }
    }
}
```

```
    }  
  }  
  else  
  {  
    // No "view" method means we are < 4.0  
    // Manually unsuspend  
    [OpenALManager sharedInstance].manuallySuspended = NO;  
  }  
}
```

### 1.11.2 MPMusicPlayerController on iOS 4.0

On iOS 4.0, MPMusicPlayerController sends an interrupt when it begins playback, but doesn't send a corresponding "end interrupt" when it ends. To work around this, force an "end interrupt" after starting playback:

```
[[OALAudioSession sharedInstance] forceEndInterruption];
```

## 1.12 Simulator Issues

As you've likely heard time and time again, the simulator is no substitute for the real thing. The simulator is buggy. It can run faster or slower than a real device. It fails system calls that a real device doesn't. It shows graphics glitches that a real device doesn't. Sounds stop working, clicks and static, dogs and cats living together, etc, etc. When things look wrong, try it on a real device before bugging people.

### 1.12.1 Simulator Limitations

The simulator does not support setting audio modes, so setting `allowHpod` or `honorSilentSwitch` in [OALAudioSession](#) will have no effect in the simulator.

### 1.12.2 Error Codes on the Simulator

From time to time, the simulator can get confused, and start spitting out spurious errors. When this happens, check on a real device to make sure it's not just a simulator issue. Usually quitting and restarting the simulator will fix it, but sometimes you may have to reboot your machine as well.

### 1.12.3 Playback Issues

The simulator is notoriously finicky when it comes to audio playback. Any number of programs you've installed on your mac can cause the simulator to stop playing bg music, or effects, or both!

Some things to check when sound stops working:

- Try resetting and restarting the simulator.



- Try restarting XCode, cleaning, and recompiling your project.
- Try rebooting your computer.
- Open "Audio MIDI Setup" (type "midi" into spotlight to find it) and make sure "Built-in Output" is set to 44100.0 Hz.
- Go to System Preferences -> Sound -> Output, and ensure that "Play sound effects through" is set to "Internal Speakers"
- Go to System Preferences -> Sound -> Input, and ensure that it is using internal sound devices.
- Go to System Preferences -> Sound -> Sound Effects, and ensure "Play user interface sound effects" is checked.
- Some codecs may cause problems with sound playback. Try removing them.
- Programs that redirect audio can wreak havoc on the simulator. Try removing them.

#### 1.12.4 No OpenAL Sound in Simulator

**Note:** As of XCode 3.2.3, this problem doesn't seem to be surfacing anymore. The workaround code is now disabled by default. You can re-enable it by setting `OBJECTAL_CFG_SIMULATOR_BUG_WORKAROUND` to 1 in [ObjectALConfig.h](#).

There's a bug in the simulator that causes OpenAL-based sounds to stop playing in certain cases when using `AVAudioPlayer` ([OALAudioTrack](#)). `ObjectAL` contains code to work around this issue, but it's not a 100% fix.

#### 1.12.5 Simulator Freezups

**Note:** As of XCode 3.2.3, this problem doesn't seem to be surfacing anymore. The workaround code is now disabled by default. You can re-enable it by setting `OBJECTAL_CFG_SIMULATOR_BUG_WORKAROUND` to 1 in [ObjectALConfig.h](#).

There's a particularly nasty bug in the simulator's OpenAL and `AVAudioPlayer` implementation that causes the simulator to freeze for 60+ seconds in a very specific case:

If you use [OALAudioTrack](#) to play background music, then stop the music, then close the current OpenAL context, the simulator will freeze (a real device won't).

This is not really a huge problem, however, since you really should be making a sound manager singleton object (what [OALSimpleAudio](#) is, basically) to handle the [ALDevice](#) and [ALContext](#) (which will in 99.9% of cases last for the entire duration of your program).

If you absolutely must close the current OpenAL context, start any [OALAudioTrack](#) objects playing at 0 volume first.



## Chapter 2

# Class Index

### 2.1 Class Hierarchy

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

ALBuffer . . . . .	21
ALCaptureDevice . . . . .	24
ALOrientation . . . . .	53
ALPoint . . . . .	54
<ALSoundSource> . . . . .	55
ALChannelSource . . . . .	29
ALSource . . . . .	66
ALSoundSourcePool . . . . .	64
ALVector . . . . .	73
ALWrapper . . . . .	74
IOSVersion . . . . .	112
NSMutableArray . . . . .	113
OAL_AsyncALBufferLoadOperation . . . . .	114
OAL_AsyncAudioTrackOperation . . . . .	116
OAL_AsyncAudioTrackPlayOperation . . . . .	118
OAL_AsyncAudioTrackPreloadOperation . . . . .	120
<OAL_GainProtocol> . . . . .	120
<OAL_PanProtocol> . . . . .	121
<OAL_PitchProtocol> . . . . .	121
<OAL_PositionProtocol> . . . . .	122
OALAction . . . . .	122
OALCallAction . . . . .	158
OALConcurrentActions . . . . .	162
OALFunctionAction . . . . .	167
OALGainAction . . . . .	171
OALPanAction . . . . .	181
OALPitchAction . . . . .	181
OALMoveByAction . . . . .	175

OALMoveToAction . . . . .	178
OALPlaceAction . . . . .	182
OALSequentialActions . . . . .	187
OALTargetedAction . . . . .	211
OALActionManager . . . . .	126
OALAudioFile . . . . .	128
<OALFunction> . . . . .	166
OALExponentialFunction . . . . .	164
OALLinearFunction . . . . .	172
OALLogarithmicFunction . . . . .	173
OALReverseFunction . . . . .	183
OALSCurveFunction . . . . .	185
OALSimpleAudio . . . . .	190
OALSuspendHandler . . . . .	203
<OALSuspendListener> . . . . .	207
<OALSuspendManager> . . . . .	209
ALContext . . . . .	37
ALDevice . . . . .	46
ALListener . . . . .	50
ALSource . . . . .	66
OALAudioSession . . . . .	133
OALAudioTrack . . . . .	140
OALAudioTracks . . . . .	155
OpenALManager . . . . .	215
OALTools . . . . .	213

## Chapter 3

# Class Index

### 3.1 Class List

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

<a href="#">ALBuffer</a> (A buffer for audio data that will be played via a SoundSource ) . . .	21
<a href="#">ALCaptureDevice</a> (*UNIMPLEMENTED FOR IOS* An OpenAL device for capturing sound data ) . . . . .	24
<a href="#">ALChannelSource</a> (A Sound source composed of other sources ) . . . . .	29
<a href="#">ALContext</a> (A context encompasses a single listener and a series of sources )	37
<a href="#">ALDevice</a> (A device is a logical mapping to an audio device through the OpenAL implementation ) . . . . .	46
<a href="#">ALListener</a> (The listener represents the user who is listening to sounds in 3D space ) . . . . .	50
<a href="#">ALOrientation</a> (Represents an orientation, consisting of an "at" vector (representing the "forward" direction), and the "up" vector (representing "up" for the subject) ) . . . . .	53
<a href="#">ALPoint</a> (Represents a 3-dimensional point for certain ObjectAL properties )	54
<a href="#">&lt;ALSoundSource&gt;</a> (Manages all properties relating to an OpenAL sound source ) . . . . .	55
<a href="#">ALSoundSourcePool</a> (A pool of sound sources, which can be fetched based on availability ) . . . . .	64
<a href="#">ALSource</a> (A source represents an object that emits sound which can be heard by a listener ) . . . . .	66
<a href="#">ALVector</a> (Represents a 3-dimensional vector for certain ObjectAL properties )	73
<a href="#">ALWrapper</a> (A thin wrapper around the C OpenAL API, with a few convenience methods thrown in ) . . . . .	74
<a href="#">IOSVersion</a> (Reports the version of iOS being run on the current device ) . . .	112
<a href="#">NSMutableArray</a> . . . . .	113
<a href="#">OAL.AsyncALBufferLoadOperation</a> ((INTERNAL USE) NSOperation for loading audio files asynchronously ) . . . . .	114
<a href="#">OAL.AsyncAudioTrackOperation</a> ((INTERNAL USE) NSOperation for running an audio operation asynchronously ) . . . . .	116

<a href="#">OAL_AsyncAudioTrackPlayOperation</a> ((INTERNAL USE) NSOperation for playing an audio file asynchronously ) . . . . .	118
<a href="#">OAL_AsyncAudioTrackPreloadOperation</a> ((INTERNAL USE) NSOperation for preloading an audio file asynchronously ) . . . . .	120
<a href="#">&lt;OAL_GainProtocol&gt;</a> ((INTERNAL USE) Protocol to keep the compiler happy )	120
<a href="#">&lt;OAL_PanProtocol&gt;</a> ((INTERNAL USE) Protocol to keep the compiler happy )	121
<a href="#">&lt;OAL_PitchProtocol&gt;</a> ((INTERNAL USE) Protocol to keep the compiler happy )	121
<a href="#">&lt;OAL_PositionProtocol&gt;</a> ((INTERNAL USE) Protocol to keep the compiler happy ) . . . . .	122
<a href="#">OALAction</a> (Represents an action that can be performed on an object ) . . . .	122
<a href="#">OALActionManager</a> (Manages all ObjectAL actions ) . . . . .	126
<a href="#">OALAudioFile</a> (Maintains an open audio file and allows loading data from that file into new <a href="#">ALBuffer</a> objects ) . . . . .	128
<a href="#">OALAudioSession</a> (Handles the audio session and interrupts ) . . . . .	133
<a href="#">OALAudioTrack</a> (Plays an audio track via AVAudioPlayer ) . . . . .	140
<a href="#">OALAudioTracks</a> (Keeps track of all AudioTrack objects ) . . . . .	155
<a href="#">OALCallAction</a> (Calls a selector on a target ) . . . . .	158
<a href="#">OALConcurrentActions</a> (A set of actions that get run concurrently ) . . . . .	162
<a href="#">OALExponentialFunction</a> (Changes slowly at the start, and quickly at the end )	164
<a href="#">&lt;OALFunction&gt;</a> (A function takes a value from 0.0 to 1.0 and returns another value from 0.0 to 1.0 ) . . . . .	166
<a href="#">OALFunctionAction</a> (An action that applies a function to the proportionComplete parameter in [update] before applying the result to the target ) . . . . .	167
<a href="#">OALGainAction</a> (A function-based action that modifies the target's gain ) . . . .	171
<a href="#">OALLinearFunction</a> (Function that changes at a constant rate ) . . . . .	172
<a href="#">OALLogarithmicFunction</a> (Changes quickly at the start, and slowly at the end )	173
<a href="#">OALMoveByAction</a> (Moves the target from its current position by the specified delta over time in 3D space ) . . . . .	175
<a href="#">OALMoveToAction</a> (Moves the target from its current position to the specified position over time in 3D space ) . . . . .	178
<a href="#">OALPanAction</a> (A function-based action that modifies the target's pan ) . . . .	181
<a href="#">OALPitchAction</a> (A function-based action that modifies the target's pitch ) . . .	181
<a href="#">OALPlaceAction</a> (Places the target at the specified position ) . . . . .	182
<a href="#">OALReverseFunction</a> (Returns the reverse of another function ) . . . . .	183
<a href="#">OALSCurveFunction</a> (Changes slowly at the start, quickly at the midpoint, then slowly again at the end ) . . . . .	185
<a href="#">OALSequentialActions</a> (A set of actions that get run in sequence ) . . . . .	187
<a href="#">OALSimpleAudio</a> (A simpler interface to the ObjectAL sound library ) . . . . .	190
<a href="#">OALSuspendHandler</a> (Provides two controls (interrupted and manuallySuspended) for suspending a slave object, and also propagates such control messages to interested listeners ) . . . . .	203
<a href="#">&lt;OALSuspendListener&gt;</a> (Allows an object to participate in interrupt and suspend operations ) . . . . .	207
<a href="#">&lt;OALSuspendManager&gt;</a> (A suspend manager is a listener that also allows other objects to subscribe to receive events as the manager receives them ) . . . . .	209
<a href="#">OALTargetedAction</a> (Ignores whatever target it was invoked upon and applies the specified action on the target specified at creation time ) . . . . .	211
<a href="#">OALTools</a> (Miscellaneous tools used by ObjectAL ) . . . . .	213

[OpenALManager](#) (Manager class for OpenAL objects (ObjectAL) ) . . . . . 215





## Chapter 4

# Class Documentation

### 4.1 ALBuffer Class Reference

A buffer for audio data that will be played via a SoundSource.

```
#import <ALBuffer.h>
```

#### Public Member Functions

- (id) - [initWithName:data:size:format:frequency:](#)  
*Initialize the buffer.*

#### Static Public Member Functions

- (id) + [bufferWithName:data:size:format:frequency:](#)  
*Make a new buffer.*

#### Protected Attributes

- void \* [bufferData](#)  
*The uncompressed sound data to play.*

#### Properties

- ALuint [bits](#)  
*The size of a sample in bits.*

- ALuint [bufferId](#)  
*The ID assigned to this buffer by OpenAL.*
- ALuint [channels](#)  
*The number of channels the buffer data plays in.*
- ALDevice \* [device](#)  
*The device this buffer was created for.*
- ALenum [format](#)  
*The format of the audio data (see *al.h*, *AL\_FORMAT\_XXX*).*
- ALuint [frequency](#)  
*The frequency this buffer runs at.*
- NSString \* [name](#)  
*The name given to this buffer upon creation.*
- ALuint [size](#)  
*The size, in bytes, of the currently loaded buffer data.*
- float [duration](#)  
*The duration of the sample in this buffer, in seconds.*
- bool [freeDataOnDestroy](#)  
*If true, calls *free()* on the audio data when this object gets destroyed.*

### 4.1.1 Detailed Description

A buffer for audio data that will be played via a SoundSource.

#### See also

SoundSource

### 4.1.2 Member Function Documentation

- 4.1.2.1 + (id) [bufferWithName:](#) *dummy(NSString\*) name data:(void\*) data size:(ALsizei) size format:(ALenum) format frequency:(ALsizei) frequency*

Make a new buffer.

#### Parameters

<i>name</i>	Optional name that you can use to identify this buffer in your code.
-------------	--

<i>data</i>	The sound data. Note: <a href="#">ALBuffer</a> will call <code>free()</code> on this data when it is destroyed!
<i>size</i>	The size of the data in bytes.
<i>format</i>	The format of the data (see the Core Audio documentation).
<i>frequency</i>	The sampling frequency in Hz.

**Returns**

A new buffer.

**4.1.2.2** - (id) initWithName: dummy(NSString\*) name data:(void\*) data size:(ALsizei) size format:(ALenum) format frequency:(ALsizei) frequency

Initialize the buffer.

**Parameters**

<i>name</i>	Optional name that you can use to identify this buffer in your code.
<i>data</i>	The sound data. Note: <a href="#">ALBuffer</a> will call <code>free()</code> on this data when it is destroyed!
<i>size</i>	The size of the data in bytes.
<i>format</i>	The format of the data (see the Core Audio documentation).
<i>frequency</i>	The sampling frequency in Hz.

**Returns**

The initialized buffer.

**4.1.3 Member Data Documentation**

**4.1.3.1** - (void\*) `bufferData` [protected]

The uncompressed sound data to play.

**4.1.4 Property Documentation**

**4.1.4.1** - (ALuint) `bits` [read, assign]

The size of a sample in bits.

**4.1.4.2** - (ALuint) `bufferId` [read, assign]

The ID assigned to this buffer by OpenAL.

#### 4.1.4.3 - (ALuint) channels [read, assign]

The number of channels the buffer data plays in.

#### 4.1.4.4 - (ALDevice \*) device [read, assign]

The device this buffer was created for.

#### 4.1.4.5 - (float) duration [read, assign]

The duration of the sample in this buffer, in seconds.

#### 4.1.4.6 - (ALenum) format [read, assign]

The format of the audio data (see al.h, AL\_FORMAT\_XXX).

#### 4.1.4.7 - (bool) freeDataOnDestroy [read, write, assign]

If true, calls free() on the audio data when this object gets destroyed.

Default: YES

#### 4.1.4.8 - (ALuint) frequency [read, assign]

The frequency this buffer runs at.

#### 4.1.4.9 - (NSString \*) name [read, write, retain]

The name given to this buffer upon creation.

You may change it at runtime if you wish.

#### 4.1.4.10 - (ALuint) size [read, assign]

The size, in bytes, of the currently loaded buffer data.

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

- ALBuffer.h
- ALBuffer.m

## 4.2 ALCaptureDevice Class Reference

\*UNIMPLEMENTED FOR IOS\* An OpenAL device for capturing sound data.

```
#import <ALCaptureDevice.h>
```

### Public Member Functions

- (id) - [initWithDeviceSpecifier:frequency:format:bufferSize:](#)  
*Open the specified device.*
- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (bool) - [startCapture](#)  
*Start capturing samples.*
- (bool) - [stopCapture](#)  
*Stop capturing samples.*
- (bool) - [moveSamples:toBuffer:](#)  
*Move captured samples to the specified buffer.*
- (bool) - [isExtensionPresent:](#)  
*Check if the specified extension is present.*
- (void \*) - [getProcAddress:](#)  
*Get the address of the specified procedure (C function address).*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*

### Static Public Member Functions

- (id) + [deviceWithDeviceSpecifier:frequency:format:bufferSize:](#)  
*Open the specified device.*

### Properties

- int [captureSamples](#)  
*The number of capture samples available.*
- ALCdevice \* [device](#)  
*The OpenAL device pointer.*
- NSArray \* [extensions](#)  
*List of strings describing all extensions available on this device (NSString\*).*

- int [majorVersion](#)

*The specification revision for this implementation (major version).*

- int [minorVersion](#)

*The specification revision for this implementation (minor version).*

## 4.2.1 Detailed Description

**\*UNIMPLEMENTED FOR IOS\*** An OpenAL device for capturing sound data. Note: This functionality is NOT implemented in iOS OpenAL!

This class is a placeholder in case such functionality is added in a future iOS SDK.

## 4.2.2 Member Function Documentation

### 4.2.2.1 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

### 4.2.2.2 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

### 4.2.2.3 + (id) deviceWithDeviceSpecifier: *dummy*(NSString\*) *deviceSpecifier* frequency:(ALCuint) *frequency* format:(ALCenum) *format* bufferSize:(ALCsizei) *bufferSize*

Open the specified device.

#### Parameters

<i>deviceSpecifier</i>	The name of the device to open (nil = default device).
<i>frequency</i>	The frequency to capture at.
<i>format</i>	The audio format to capture as.
<i>bufferSize</i>	The size of buffer that the device must allocate for audio capture.

#### Returns

A new capture device.

**4.2.2.4 - (void \*) getProcAddress: dummy(NSString\*) functionName**

Get the address of the specified procedure (C function address).

**Parameters**

<i>function-Name</i>	The name of the procedure to get.
----------------------	-----------------------------------

**Returns**

the procedure's address, or NULL if it wasn't found.

**4.2.2.5 - (id) initWithDeviceSpecifier: dummy(NSString\*) deviceSpecifier frequency:(ALCuint) frequency format:(ALCenum) format bufferSize:(ALCsizei) bufferSize**

Open the specified device.

**Parameters**

<i>deviceSpecifier</i>	The name of the device to open (nil = default device).
<i>frequency</i>	The frequency to capture at.
<i>format</i>	The audio format to capture as.
<i>bufferSize</i>	The size of buffer that the device must allocate for audio capture.

**Returns**

The initialized capture device.

**4.2.2.6 - (bool) isExtensionPresent: dummy(NSString\*) name**

Check if the specified extension is present.

**Parameters**

<i>name</i>	The name of the extension to check.
-------------	-------------------------------------

**Returns**

TRUE if the extension is present.

**4.2.2.7 - (bool) moveSamples: dummy(ALCsizei) numSamples toBuffer:(ALCvoid\*) buffer**

Move captured samples to the specified buffer.

This method will fail if less than the specified number of samples have been captured.

**Parameters**

<i>numSamples</i>	The number of samples to move.
<i>buffer</i>	the buffer to move the samples into.

**Returns**

TRUE if the operation was successful.

**4.2.2.8 - (bool) startCapture**

Start capturing samples.

**Returns**

TRUE if the operation was successful.

**4.2.2.9 - (bool) stopCapture**

Stop capturing samples.

**Returns**

TRUE if the operation was successful.

**4.2.3 Property Documentation****4.2.3.1 - (int) captureSamples** [read, assign]

The number of capture samples available.

**4.2.3.2 - (ALCdevice \*) device** [read, assign]

The OpenAL device pointer.

**4.2.3.3 - (NSArray \*) extensions** [read, assign]

List of strings describing all extensions available on this device (NSString\*).

**4.2.3.4 - (int) majorVersion** [read, assign]

The specification revision for this implementation (major version).



#### 4.2.3.5 - (int) minorVersion [read, assign]

The specification revision for this implementation (minor version).

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

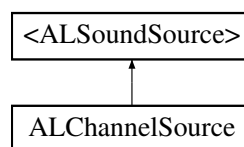
- `ALCaptureDevice.h`
- `ALCaptureDevice.m`

## 4.3 ALChannelSource Class Reference

A Sound source composed of other sources.

```
#import <ALChannelSource.h>
```

Inheritance diagram for ALChannelSource:



### Public Member Functions

- (id) - [initWithSources:](#)  
*Initialize a channel with a number of sources.*
- (void) - [resetToDefault](#)  
*Reset all sources in this channel to their default state.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*
- (void) - [onFadeComplete:](#)  
*(INTERNAL USE) Called by the action system when a fade completes.*
- (void) - [onPanComplete:](#)  
*(INTERNAL USE) Called by the action system when a pan completes.*
- (void) - [onPitchComplete:](#)  
*(INTERNAL USE) Called by the action system when a pitch change completes.*

## Static Public Member Functions

- (id) + [channelWithSources](#):  
*Create a channel with a number of sources.*

## Protected Attributes

- float [pitch](#)  
*Pitch (OpenAL property).*
- float [gain](#)  
*Gain (volume) (OpenAL property).*
- float [maxDistance](#)  
*Max distance (OpenAL property).*
- float [rolloffFactor](#)  
*Rolloff factor (OpenAL property).*
- float [referenceDistance](#)  
*Reference distance (OpenAL property).*
- float [minGain](#)  
*Min gain (OpenAL property).*
- float [maxGain](#)  
*Max gain (OpenAL property).*
- float [coneOuterGain](#)  
*Cone outer gain (OpenAL property).*
- float [coneInnerAngle](#)  
*Cone inner angle (OpenAL property).*
- float [coneOuterAngle](#)  
*Cone outer angle (OpenAL property).*
- [ALPoint](#) [position](#)  
*Position (OpenAL property).*
- [ALVector](#) [velocity](#)  
*Velocity (OpenAL property).*
- [ALVector](#) [direction](#)  
*Direction (OpenAL property).*

- int [sourceRelative](#)  
*Source relative (OpenAL property).*
- int [sourceType](#)  
*Source type (OpenAL property).*
- bool [looping](#)  
*Looping (OpenAL property).*
- bool [interruptible](#)  
*If true, this source may be interrupted when resources are low.*
- bool [muted](#)  
*If true, this source is muted.*
- bool [paused](#)  
*If true, this source is currently paused.*
- id [fadeCompleteTarget](#)  
*Target to inform when the current fade operation completes.*
- SEL [fadeCompleteSelector](#)  
*Selector to call when the current fade operation completes.*
- int [expectedFadeCallbackCount](#)  
*The expected number of sources that will callback when fading completes.*
- int [currentFadeCallbackCount](#)  
*The actual number of sources that have called back.*
- id [panCompleteTarget](#)  
*Target to inform when the current pan operation completes.*
- SEL [panCompleteSelector](#)  
*Selector to call when the current pan operation completes.*
- int [expectedPanCallbackCount](#)  
*The expected number of sources that will callback when panning completes.*
- int [currentPanCallbackCount](#)  
*The actual number of sources that have called back.*
- id [pitchCompleteTarget](#)  
*Target to inform when the current pitch operation completes.*

- SEL [pitchCompleteSelector](#)  
*Selector to call when the current pitch operation completes.*
- int [expectedPitchCallbackCount](#)  
*The expected number of sources that will callback when pitch op completes.*
- int [currentPitchCallbackCount](#)  
*The actual number of sources that have called back.*

## Properties

- [ALContext](#) \* [context](#)  
*This source's owning context.*
- [ALSoundSourcePool](#) \* [sourcePool](#)  
*All sources being used by this channel.*
- unsigned int [reservedSources](#)  
*The number of sources reserved by this channel.*

### 4.3.1 Detailed Description

A Sound source composed of other sources. Property values are applied to all sources within the channel.

Sounds will get played by any free sources within this channel.

If all sources are busy when playback is requested, it will attempt to interrupt a source to free it for playback.

### 4.3.2 Member Function Documentation

#### 4.3.2.1 + (id) channelWithSources: dummy(int) *reservedSources*

Create a channel with a number of sources.

#### Parameters

<i>reserved-Sources</i>	the number of sources to reserve for this channel.
-------------------------	--

#### Returns

A new channel.

**4.3.2.2 - (void) closeOSResources**

(INTERNAL USE) Close any resources belonging to the OS.

**4.3.2.3 - (id) initWithSources: dummy(int) reservedSources**

Initialize a channel with a number of sources.

**Parameters**

<i>reserved-Sources</i>	the number of sources to reserve for this channel.
-------------------------	--

**Returns**

The initialized channel.

**4.3.2.4 - (void) onFadeComplete: dummy(id< ALSoundSource >) source**

(INTERNAL USE) Called by the action system when a fade completes.

**4.3.2.5 - (void) onPanComplete: dummy(id< ALSoundSource >) source**

(INTERNAL USE) Called by the action system when a pan completes.

**4.3.2.6 - (void) onPitchComplete: dummy(id< ALSoundSource >) source**

(INTERNAL USE) Called by the action system when a pitch change completes.

**4.3.2.7 - (void) resetToDefault**

Reset all sources in this channel to their default state.

**4.3.3 Member Data Documentation****4.3.3.1 - (float) coneInnerAngle** [protected]

Cone inner angle (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.2 - (float) coneOuterAngle** [protected]

Cone outer angle (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.3 - (float) coneOuterGain** [protected]

Cone outer gain (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.4 - (int) currentFadeCallbackCount** [protected]

The actual number of sources that have called back.

**4.3.3.5 - (int) currentPanCallbackCount** [protected]

The actual number of sources that have called back.

**4.3.3.6 - (int) currentPitchCallbackCount** [protected]

The actual number of sources that have called back.

**4.3.3.7 - (ALVector) direction** [protected]

Direction (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.8 - (int) expectedFadeCallbackCount** [protected]

The expected number of sources that will callback when fading completes.

**4.3.3.9 - (int) expectedPanCallbackCount** [protected]

The expected number of sources that will callback when panning completes.

**4.3.3.10 - (int) expectedPitchCallbackCount** [protected]

The expected number of sources that will callback when pitch op completes.

**4.3.3.11 - (SEL) fadeCompleteSelector** [protected]

Selector to call when the current fade operation completes.

**4.3.3.12 - (id) fadeCompleteTarget** [protected]

Target to inform when the current fade operation completes.

**4.3.3.13 - (float) gain** [protected]

Gain (volume) (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.14 - (bool) interruptible** [protected]

If true, this source may be interrupted when resources are low.

Reimplemented from [<ALSoundSource>](#).

**4.3.3.15 - (bool) looping** [protected]

Looping (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.16 - (float) maxDistance** [protected]

Max distance (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.17 - (float) maxGain** [protected]

Max gain (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.18 - (float) minGain** [protected]

Min gain (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.19 - (bool) muted** [protected]

If true, this source is muted.

Reimplemented from [<ALSoundSource>](#).

**4.3.3.20 - (SEL) panCompleteSelector** [protected]

Selector to call when the current pan operation completes.

**4.3.3.21 - (id) panCompleteTarget** [protected]

Target to inform when the current pan operation completes.

**4.3.3.22 - (bool) paused** [protected]

If true, this source is currently paused.

Reimplemented from [<ALSoundSource>](#).

**4.3.3.23 - (float) pitch** [protected]

Pitch (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.24 - (SEL) pitchCompleteSelector** [protected]

Selector to call when the current pitch operation completes.

**4.3.3.25 - (id) pitchCompleteTarget** [protected]

Target to inform when the current pitch operation completes.

**4.3.3.26 - (ALPoint) position** [protected]

Position (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.27 - (float) referenceDistance** [protected]

Reference distance (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.28 - (float) rolloffFactor** [protected]

Rolloff factor (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.29 - (int) sourceRelative** [protected]

Source relative (OpenAL property).

Reimplemented from [<ALSoundSource>](#).



**4.3.3.30** - (int) `sourceType` [protected]

Source type (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.3.31** - (ALVector) `velocity` [protected]

Velocity (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.3.4 Property Documentation****4.3.4.1** - (ALContext \*) `context` [read, assign]

This source's owning context.

**4.3.4.2** - (unsigned int) `reservedSources` [read, write, assign]

The number of sources reserved by this channel.

**4.3.4.3** - (ALSoundSourcePool \*) `sourcePool` [read, assign]

All sources being used by this channel.

Do not modify!

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

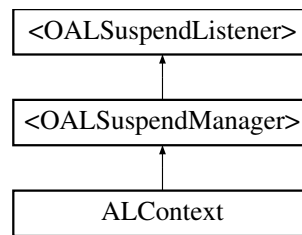
- `ALChannelSource.h`
- `ALChannelSource.m`

**4.4 ALContext Class Reference**

A context encompasses a single listener and a series of sources.

```
#import <ALContext.h>
```

Inheritance diagram for ALContext:



## Public Member Functions

- (id) - [initOnDevice:outputFrequency:refreshIntervals:synchronousContext:monoSources:stereoSources:](#)  
*Initialize this context on the specified device with attributes.*
- (id) - [initOnDevice:attributes:](#)  
*Initialize this context for the specified device and attributes.*
- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (void) - [process](#)  
*Process this context.*
- (void) - [stopAllSounds](#)  
*Stop all sound sources in this context.*
- (void) - [clearBuffers](#)  
*Clear all buffers being used by sources in this context.*
- (void) - [ensureContextIsCurrent](#)  
*Make sure this context is the current context.*
- (bool) - [isExtensionPresent:](#)  
*Check if the specified extension is present in this context.*
- (void \*) - [getProcAddress:](#)  
*Get the address of the specified procedure (C function address).*
- (void) - [notifySourceInitializing:](#)  
*(INTERNAL USE) Used by [ALSource](#) to announce initialization.*
- (void) - [notifySourceDeallocating:](#)  
*(INTERNAL USE) Used by [ALSource](#) to announce deallocation.*
- (void) - [closeOSResources](#)

*(INTERNAL USE) Close any resources belonging to the OS.*

- (void) - [setSuspended](#):  
*(INTERNAL USE) Called by SuspendHandler.*

### Static Public Member Functions

- (id) + [contextOnDevice:attributes](#):  
*Create a new context on the specified device.*
- (id) + [contextOnDevice:outputFrequency:refreshIntervals:synchronousContext:monoSources:stereoSources](#):  
*Create a new context on the specified device with attributes.*

### Protected Attributes

- [NSMutableArray](#) \* [sources](#)  
*All sound sources associated with this context.*
- bool [suspended](#)  
*If YES, this object is suspended.*
- [NSMutableArray](#) \* [attributes](#)  
*This context's attributes.*
- [OALSuspendHandler](#) \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*

### Properties

- NSString \* [alVersion](#)  
*OpenAL version string in format "[spec major number]".*
- NSArray \* [attributes](#)  
*The current context's attribute list.*
- ALCcontext \* [context](#)  
*The OpenAL context pointer.*
- [ALDevice](#) \* [device](#)  
*The device this context was opened on.*

- AEnum [distanceModel](#)  
*The current distance model.*
- float [dopplerFactor](#)  
*Exaggeration factor for Doppler effect.*
- NSArray \* [extensions](#)  
*List of available extensions (NSString\*).*
- ALListener \* [listener](#)  
*This context's listener.*
- NSString \* [renderer](#)  
*Information about the specific renderer.*
- NSArray \* [sources](#)  
*All sources associated with this context (ALSource\*).*
- float [speedOfSound](#)  
*Speed of sound in same units as velocities.*
- NSString \* [vendor](#)  
*Name of the vendor.*

#### 4.4.1 Detailed Description

A context encompasses a single listener and a series of sources. A context is created from a device, and many contexts may be created (though multiple contexts would be unusual in an iOS app).

Note: Some property values are only valid if this context is the current context.

##### See also

`ObjectAL.currentContext`

#### 4.4.2 Member Function Documentation

##### 4.4.2.1 - (void) clearBuffers

Clear all buffers being used by sources in this context.

##### 4.4.2.2 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

**4.4.2.3 - (void) closeOSResources**

(INTERNAL USE) Close any resources belonging to the OS.

**4.4.2.4 + (id) contextOnDevice: dummy(ALDevice \*) device attributes:(NSArray\*) attributes**

Create a new context on the specified device.

**Parameters**

<i>device</i>	The device to open the context on.
<i>attributes</i>	An array of NSNumber in ordered pairs (attribute id followed by integer value). Possible attributes: ALC_FREQUENCY, ALC_REFRESH, ALC_SYNC, ALC_MONO_SOURCES, ALC_STEREO_SOURCES

**Returns**

A new context.

**4.4.2.5 + (id) contextOnDevice: dummy(ALDevice\*) device outputFrequency:(int) outputFrequency refreshIntervals:(int) refreshIntervals synchronousContext:(bool) synchronousContext monoSources:(int) monoSources stereoSources:(int) stereoSources**

Create a new context on the specified device with attributes.

**Parameters**

<i>device</i>	The device to open the context on.
<i>outputFrequency</i>	The frequency to mix all sources to before outputting.
<i>refreshIntervals</i>	The number of passes per second used to mix the audio sources. For games this can be 5-15. For audio intensive apps, it should be higher.
<i>synchronousContext</i>	If true, this context runs on the main thread and depends on you calling alcUpdateContext (best to leave this FALSE unless you know what you're doing).
<i>monoSources</i>	A hint indicating how many sources should support mono.
<i>stereoSources</i>	A hint indicating how many sources should support stereo.

**Returns**

A new context.

**4.4.2.6 - (void) ensureContextIsCurrent**

Make sure this context is the current context.

This method is used to work around iOS 4.0 and 4.2 bugs that could cause the context to be lost.

#### 4.4.2.7 - (void \*) **getProcAddress:** *dummy*(NSString\*) *functionName*

Get the address of the specified procedure (C function address).

Only valid when this is the current context.

**Note:** The OpenAL implementation is free to return a pointer even if it is not valid for this context. Always call `isExtensionPresent` first.

##### Parameters

<i>function-Name</i>	the name of the procedure to get.
----------------------	-----------------------------------

##### Returns

the procedure's address, or NULL if it wasn't found.

#### 4.4.2.8 - (id) **initWithDevice:** *dummy*(ALDevice\*) *device* *attributes:(NSArray\*) attributes*

Initialize this context for the specified device and attributes.

##### Parameters

<i>device</i>	The device to open the context on.
<i>attributes</i>	An array of NSNumber in ordered pairs (attribute id followed by integer value). Possible attributes: ALC_FREQUENCY, ALC_REFRESH, ALC_SYNC, ALC_MONO_SOURCES, ALC_STEREO_SOURCES

##### Returns

The initialized context.

#### 4.4.2.9 - (id) **initWithDevice:** *dummy*(ALDevice\*) *device* *outputFrequency:(int) outputFrequency* *refreshIntervals:(int) refreshIntervals* *synchronousContext:(bool) synchronousContext* *monoSources:(int) monoSources* *stereoSources:(int) stereoSources*

Initialize this context on the specified device with attributes.

##### Parameters

<i>device</i>	The device to open the context on.
<i>outputFrequency</i>	The frequency to mix all sources to before outputting.
<i>refreshIntervals</i>	The number of passes per second used to mix the audio sources. For games this can be 5-15. For audio intensive apps, it should be higher.

<i>syn-chronous-Context</i>	If true, this context runs on the main thread and depends on you calling <code>alcUpdateContext</code> (best to leave this FALSE unless you know what you're doing).
<i>monoSources</i>	A hint indicating how many sources should support mono.
<i>stereoSources</i>	A hint indicating how many sources should support stereo.

**Returns**

The initialized context.

**4.4.2.10 - (bool) isExtensionPresent: dummy(NSString\*) name**

Check if the specified extension is present in this context.

Only valid when this is the current context.

**Parameters**

<i>name</i>	The name of the extension to check.
-------------	-------------------------------------

**Returns**

TRUE if the extension is present in this context.

**4.4.2.11 - (void) notifySourceDeallocating: dummy(ALSource\*) source**

(INTERNAL USE) Used by [ALSource](#) to announce deallocation.

**Parameters**

<i>source</i>	the source that is deallocating.
---------------	----------------------------------

**4.4.2.12 - (void) notifySourceInitializing: dummy(ALSource\*) source**

(INTERNAL USE) Used by [ALSource](#) to announce initialization.

**Parameters**

<i>source</i>	the source that is initializing.
---------------	----------------------------------

**4.4.2.13 - (void) process**

Process this context.

#### 4.4.2.14 - (void) setSuspended: dummy(bool) value

(INTERNAL USE) Called by SuspendHandler.

#### 4.4.2.15 - (void) stopAllSounds

Stop all sound sources in this context.

### 4.4.3 Member Data Documentation

#### 4.4.3.1 - (NSMutableArray\*) attributes [protected]

This context's attributes.

#### 4.4.3.2 - (NSMutableArray\*) sources [protected]

All sound sources associated with this context.

#### 4.4.3.3 - (bool) suspended [protected]

If YES, this object is suspended.

Reimplemented from [<OALSuspendManager>](#).

#### 4.4.3.4 - (OALSuspendHandler\*) suspendHandler [protected]

Handles suspending and interrupting for this object.

### 4.4.4 Property Documentation

#### 4.4.4.1 - (NSString \*) alVersion [read, assign]

OpenAL version string in format "[spec major number].

[spec minor number] [optional vendor version information]" Only valid when this is the current context.

#### 4.4.4.2 - (NSArray\*) attributes [read, assign]

The current context's attribute list.

Only valid when this is the current context.



**4.4.4.3 - (ALContext \*) context** [read, assign]

The OpenAL context pointer.

**4.4.4.4 - (ALDevice \*) device** [read, assign]

The device this context was opened on.

**4.4.4.5 - (ALenum) distanceModel** [read, write, assign]

The current distance model.

Legal values are AL\_NONE, AL\_INVERSE\_DISTANCE, AL\_INVERSE\_DISTANCE\_CLAMPED, AL\_LINEAR\_DISTANCE, AL\_LINEAR\_DISTANCE\_CLAMPED, AL\_EXPONENT\_DISTANCE, and AL\_EXPONENT\_DISTANCE\_CLAMPED. See the OpenAL spec for detailed information.

Only valid when this is the current context.

**4.4.4.6 - (float) dopplerFactor** [read, write, assign]

Exaggeration factor for Doppler effect.

Only valid when this is the current context.

**4.4.4.7 - (NSArray \*) extensions** [read, assign]

List of available extensions (NSString\*).

Only valid when this is the current context.

**4.4.4.8 - (ALListener \*) listener** [read, assign]

This context's listener.

**4.4.4.9 - (NSString \*) renderer** [read, assign]

Information about the specific renderer.

Only valid when this is the current context.

**4.4.4.10 - (NSArray\*) sources** [read, assign]

All sources associated with this context (ALSource\*).

#### 4.4.4.11 - (float) speedOfSound [read, write, assign]

Speed of sound in same units as velocities.

Only valid when this is the current context.

#### 4.4.4.12 - (NSString \*) vendor [read, assign]

Name of the vendor.

Only valid when this is the current context.

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

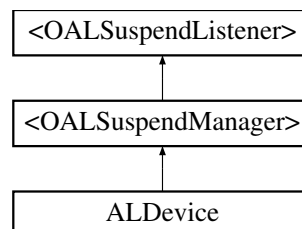
- ALContext.h
- ALContext.m

## 4.5 ALDevice Class Reference

A device is a logical mapping to an audio device through the OpenAL implementation.

```
#import <ALDevice.h>
```

Inheritance diagram for ALDevice:



### Public Member Functions

- (id) - [initWithDeviceSpecifier:](#)  
*Initialize with the specified device.*
- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (bool) - [isExtensionPresent:](#)  
*Check if the specified extension is present.*
- (void \*) - [getProcAddress:](#)  
*Get the address of the specified procedure (C function address).*

- (void) - [clearBuffers](#)  
*Clear all buffers being used by sources of contexts opened on this device.*
- (void) - [notifyContextInitializing](#):  
*(INTERNAL USE) Used by [ALContext](#) to announce initialization.*
- (void) - [notifyContextDeallocating](#):  
*(INTERNAL USE) Used by [ALContext](#) to announce deallocation.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*

### Static Public Member Functions

- (id) + [deviceWithDeviceSpecifier](#):  
*Open the specified device.*

### Protected Attributes

- [NSMutableArray](#) \* [contexts](#)  
*All contexts opened from this device.*
- [OALSuspendHandler](#) \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*

### Properties

- [NSArray](#) \* [contexts](#)  
*All contexts created on this device ([ALContext](#)\*).*
- [ALCdevice](#) \* [device](#)  
*The OpenAL device pointer.*
- [NSArray](#) \* [extensions](#)  
*List of strings describing all extensions available on this device ([NSString](#)\*).*
- int [majorVersion](#)  
*The specification revision for this implementation (major version).*
- int [minorVersion](#)  
*The specification revision for this implementation (minor version).*

### 4.5.1 Detailed Description

A device is a logical mapping to an audio device through the OpenAL implementation.

### 4.5.2 Member Function Documentation

#### 4.5.2.1 - (void) clearBuffers

Clear all buffers being used by sources of contexts opened on this device.

#### 4.5.2.2 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

#### 4.5.2.3 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

#### 4.5.2.4 + (id) deviceWithDeviceSpecifier: *dummy*(NSString\*) *deviceSpecifier*

Open the specified device.

##### Parameters

<i>deviceSpecifier</i>	The device to open (nil = default device).
------------------------	--

##### Returns

A new device.

#### 4.5.2.5 - (void \*) getProcAddress: *dummy*(NSString\*) *functionName*

Get the address of the specified procedure (C function address).

##### Parameters

<i>functionName</i>	the name of the procedure to get.
---------------------	-----------------------------------

##### Returns

the procedure's address, or NULL if it wasn't found.

**4.5.2.6 - (id) initWithDeviceSpecifier: dummy(NSString\*) deviceSpecifier**

Initialize with the specified device.

**Parameters**

<i>deviceSpecifier</i>	The device to open (nil = default device).
------------------------	--

**Returns**

the initialized device.

**4.5.2.7 - (bool) isExtensionPresent: dummy(NSString\*) name**

Check if the specified extension is present.

**Parameters**

<i>name</i>	The extension to check.
-------------	-------------------------

**Returns**

TRUE if the extension is present.

**4.5.2.8 - (void) notifyContextDeallocating: dummy(ALContext\*) context**

(INTERNAL USE) Used by [ALContext](#) to announce deallocation.

**Parameters**

<i>context</i>	The context that is deallocating.
----------------	-----------------------------------

**4.5.2.9 - (void) notifyContextInitializing: dummy(ALContext\*) context**

(INTERNAL USE) Used by [ALContext](#) to announce initialization.

**Parameters**

<i>context</i>	The context that is initializing.
----------------	-----------------------------------

**4.5.3 Member Data Documentation****4.5.3.1 - (NSMutableArray\*) contexts [protected]**

All contexts opened from this device.

#### 4.5.3.2 - (OALSuspendHandler\*) suspendHandler [protected]

Handles suspending and interrupting for this object.

### 4.5.4 Property Documentation

#### 4.5.4.1 - (NSArray\*) contexts [read, assign]

All contexts created on this device (ALContext\*).

#### 4.5.4.2 - (ALCdevice \*) device [read, assign]

The OpenAL device pointer.

#### 4.5.4.3 - (NSArray \*) extensions [read, assign]

List of strings describing all extensions available on this device (NSString\*).

#### 4.5.4.4 - (int) majorVersion [read, assign]

The specification revision for this implementation (major version).

#### 4.5.4.5 - (int) minorVersion [read, assign]

The specification revision for this implementation (minor version).

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

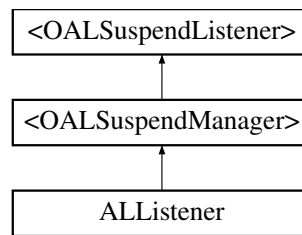
- ALDevice.h
- ALDevice.m

## 4.6 ALListener Class Reference

The listener represents the user who is listening to sounds in 3D space.

```
#import <ALListener.h>
```

Inheritance diagram for ALListener:



### Public Member Functions

- (id) - [initWithContext:](#)  
*(INTERNAL USE) Initialize a listener for the specified context.*

### Static Public Member Functions

- (id) + [listenerForContext:](#)  
*(INTERNAL USE) Create a listener for the specified context.*

### Protected Attributes

- [OALSuspendHandler](#) \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*

### Properties

- [ALContext](#) \* [context](#)  
*The context this listener belongs to.*
- bool [muted](#)  
*Causes this listener to stop hearing sound.*
- float [gain](#)  
*Gain (volume), affecting every sound this listener hears (0.0 = no sound, 1.0 = max volume).*
- [ALOrientation](#) [orientation](#)  
*Orientation (up: x, y, z, at: x, y, z).*
- [ALPoint](#) [position](#)  
*Position (x, y, z).*

- [ALVector velocity](#)  
*Velocity (x, y, z).*

#### 4.6.1 Detailed Description

The listener represents the user who is listening to sounds in 3D space. This object controls his position, orientation, and velocity, as well as providing a master gain.

A context contains one and only one listener.

#### 4.6.2 Member Function Documentation

##### 4.6.2.1 - (id) initWithContext: dummy(ALContext\*) context

(INTERNAL USE) Initialize a listener for the specified context.

###### Parameters

<i>context</i>	the context to create this listener on.
----------------	---

###### Returns

The initialized listener.

##### 4.6.2.2 + (id) listenerForContext: dummy(ALContext\*) context

(INTERNAL USE) Create a listener for the specified context.

###### Parameters

<i>context</i>	the context to create this listener on.
----------------	---

###### Returns

A new listener.

#### 4.6.3 Member Data Documentation

##### 4.6.3.1 - (OALSuspendHandler\*) suspendHandler [protected]

Handles suspending and interrupting for this object.



## 4.6.4 Property Documentation

### 4.6.4.1 `-(ALContext *) context` [read, assign]

The context this listener belongs to.

### 4.6.4.2 `-(float) gain` [read, write, assign]

Gain (volume), affecting every sound this listener hears (0.0 = no sound, 1.0 = max volume).

Only valid if this listener's context is the current context.

### 4.6.4.3 `-(bool) muted` [read, write, assign]

Causes this listener to stop hearing sound.

It's called "muted" rather than "deaf" to give a consistent name with other mute functions.

### 4.6.4.4 `-(ALOrientation) orientation` [read, write, assign]

Orientation (up: x, y, z, at: x, y, z).

Only valid if this listener's context is the current context.

### 4.6.4.5 `-(ALPoint) position` [read, write, assign]

Position (x, y, z).

Only valid if this listener's context is the current context.

### 4.6.4.6 `-(ALVector) velocity` [read, write, assign]

Velocity (x, y, z).

Only valid if this listener's context is the current context.

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

- ALLlistener.h
- ALLlistener.m

## 4.7 ALOrientation Struct Reference

Represents an orientation, consisting of an "at" vector (representing the "forward" direction), and the "up" vector (representing "up" for the subject).

```
#include <ALTypes.h>
```

### Public Attributes

- [ALVector at](#)  
*The "at" vector, representing "forward".*
- [ALVector up](#)  
*The "up" vector, representing "up".*

#### 4.7.1 Detailed Description

Represents an orientation, consisting of an "at" vector (representing the "forward" direction), and the "up" vector (representing "up" for the subject).

#### 4.7.2 Member Data Documentation

##### 4.7.2.1 [ALVector ALOrientation::at](#)

The "at" vector, representing "forward".

##### 4.7.2.2 [ALVector ALOrientation::up](#)

The "up" vector, representing "up".

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

- [ALTypes.h](#)

### 4.8 ALPoint Struct Reference

Represents a 3-dimensional point for certain ObjectAL properties.

```
#include <ALTypes.h>
```

### Public Attributes

- float [x](#)  
*The "X" coordinate.*
- float [y](#)  
*The "Y" coordinate.*

- float [z](#)

*The "Z" coordinate.*

### 4.8.1 Detailed Description

Represents a 3-dimensional point for certain ObjectAL properties.

### 4.8.2 Member Data Documentation

#### 4.8.2.1 float ALPoint::x

The "X" coordinate.

#### 4.8.2.2 float ALPoint::y

The "Y" coordinate.

#### 4.8.2.3 float ALPoint::z

The "Z" coordinate.

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

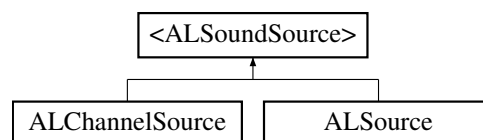
- ALTypes.h

## 4.9 <ALSoundSource> Protocol Reference

Manages all properties relating to an OpenAL sound source.

```
#import <ALSoundSource.h>
```

Inheritance diagram for <ALSoundSource>:



### Public Member Functions

- (void) - [close](#)

*Close any OS resources in use by this object.*

- (id< [ALSoundSource](#) >) - [play](#):  
*Play a sound.*
- (id< [ALSoundSource](#) >) - [play:loop](#):  
*Play a sound, optionally looping.*
- (id< [ALSoundSource](#) >) - [play:gain:pitch:pan:loop](#):  
*Play a sound, setting gain, pitch, pan, and looping.*
- (void) - [stop](#)  
*Stop playing the current sound.*
- (void) - [rewind](#)  
*Stop playing the current sound and set its state to `AL_INITIAL`.*
- (void) - [fadeTo:duration:target:selector](#):  
*Fade to the specified gain value.*
- (void) - [stopFade](#)  
*Stop the currently running fade operation, if any.*
- (void) - [panTo:duration:target:selector](#):  
*pan to the specified value.*
- (void) - [stopPan](#)  
*Stop the currently running pan operation, if any.*
- (void) - [pitchTo:duration:target:selector](#):  
*Gradually change pitch to the specified value.*
- (void) - [stopPitch](#)  
*Stop the currently running pitch operation, if any.*
- (void) - [stopActions](#)  
*Stop any currently running fade, pan, or pitch operations.*
- (void) - [clear](#)  
*Clear any buffers this source is currently using.*

## Properties

- float [coneInnerAngle](#)  
*Cone inner angle (OpenAL property).*

- float [coneOuterAngle](#)  
*Cone outer angle (OpenAL property).*
- float [coneOuterGain](#)  
*Cone outer gain (OpenAL property).*
- [ALVector direction](#)  
*Direction (OpenAL property).*
- float [gain](#)  
*Gain (volume) (OpenAL property).*
- float [volume](#)  
*Volume (alias to gain).*
- bool [interruptible](#)  
*If true, this source may be interrupted when resources are low.*
- bool [looping](#)  
*Looping (OpenAL property).*
- float [maxDistance](#)  
*Max distance (OpenAL property).*
- float [maxGain](#)  
*Max gain (OpenAL property).*
- float [minGain](#)  
*Min gain (OpenAL property).*
- bool [muted](#)  
*If true, this source is muted.*
- bool [paused](#)  
*If true, this source is currently paused.*
- float [pitch](#)  
*Pitch (OpenAL property).*
- bool [playing](#)  
*If true, this source is currently playing audio.*
- [ALPoint position](#)  
*Position (OpenAL property).*
- float [referenceDistance](#)

*Reference distance (OpenAL property).*

- float [rolloffFactor](#)  
*Rolloff factor (OpenAL property).*
- int [sourceRelative](#)  
*Source relative (OpenAL property).*
- int [sourceType](#)  
*Source type (OpenAL property).*
- [ALVector](#) [velocity](#)  
*Velocity (OpenAL property).*
- float [pan](#)  
*Pan value (-1.0 = far left, 1.0 = far right).*

#### 4.9.1 Detailed Description

Manages all properties relating to an OpenAL sound source. There are currently two classes that adhere to this protocol: [ALSource](#) and [ChannelSource](#) (which collectively manipulates a set of [ALSource](#) objects). A full description of the properties themselves is available in the OpenAL 1.1 Specification and Reference: <http://connect.creativelabs.com/openal>

#### 4.9.2 Member Function Documentation

##### 4.9.2.1 - (void) clear

Clear any buffers this source is currently using.

##### 4.9.2.2 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

##### 4.9.2.3 - (void) fadeTo: [dummy\(float\)](#) *gain* duration:(float) *duration* target:(id) *target* selector:(SEL) *selector*

Fade to the specified gain value.

##### Parameters

<i>gain</i>	The gain to fade to.
<i>duration</i>	The duration of the fade operation in seconds.

<i>target</i>	The target to notify when the fade completes (can be nil).
<i>selector</i>	The selector to call when the fade completes. The selector must accept a single parameter, which will be the object that performed the fade.

#### 4.9.2.4 - (void) panTo: dummy(float) *pan* duration:(float) *duration* target:(id) *target* selector:(SEL) *selector*

pan to the specified value.

##### Parameters

<i>pan</i>	The value to pan to.
<i>duration</i>	The duration of the pan operation in seconds.
<i>target</i>	The target to notify when the pan completes (can be nil).
<i>selector</i>	The selector to call when the pan completes. The selector must accept a single parameter, which will be the object that performed the pan.

#### 4.9.2.5 - (void) pitchTo: dummy(float) *pitch* duration:(float) *duration* target:(id) *target* selector:(SEL) *selector*

Gradually change pitch to the specified value.

##### Parameters

<i>pitch</i>	The value to change pitch to.
<i>duration</i>	The duration of the pitch operation in seconds.
<i>target</i>	The target to notify when the pitch change completes (can be nil).
<i>selector</i>	The selector to call when the pitch change completes. The selector must accept a single parameter, which will be the object that performed the pitch change.

#### 4.9.2.6 - (id<ALSoundSource>) play: dummy(ALBuffer \*) *buffer*

Play a sound.

##### Parameters

<i>buffer</i>	the buffer to play.
---------------	---------------------

##### Returns

the source playing the sound, or nil if the sound could not be played.

#### 4.9.2.7 - (id<ALSoundSource>) play: dummy(ALBuffer \*) *buffer* gain:(float) *gain* pitch:(float) *pitch* pan:(float) *pan* loop:(bool) *loop*

Play a sound, setting gain, pitch, pan, and looping.

##### Parameters

<i>buffer</i>	the buffer to play.
<i>gain</i>	The gain (volume) to play at (0.0 - 1.0).
<i>pitch</i>	The pitch to play at (1.0 = normal pitch).
<i>pan</i>	Left-right panning (-1.0 = far left, 1.0 = far right).
<i>loop</i>	If TRUE, the sound will loop until you call "stop" on the returned sound source.

##### Returns

the source playing the sound, or nil if the sound could not be played.

#### 4.9.2.8 - (id<ALSoundSource>) play: dummy(ALBuffer \*) *buffer* loop:(bool) *loop*

Play a sound, optionally looping.

##### Parameters

<i>buffer</i>	the buffer to play.
<i>loop</i>	If TRUE, the sound will loop until you call "stop" on the returned sound source.

##### Returns

the source playing the sound, or nil if the sound could not be played.

#### 4.9.2.9 - (void) rewind

Stop playing the current sound and set its state to AL\_INITIAL.

#### 4.9.2.10 - (void) stop

Stop playing the current sound.

#### 4.9.2.11 - (void) stopActions

Stop any currently running fade, pan, or pitch operations.

#### 4.9.2.12 - (void) stopFade

Stop the currently running fade operation, if any.



**4.9.2.13 - (void) stopPan**

Stop the currently running pan operation, if any.

**4.9.2.14 - (void) stopPitch**

Stop the currently running pitch operation, if any.

**4.9.3 Property Documentation****4.9.3.1 - (float) coneInnerAngle** [read, write, assign]

Cone inner angle (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.2 - (float) coneOuterAngle** [read, write, assign]

Cone outer angle (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.3 - (float) coneOuterGain** [read, write, assign]

Cone outer gain (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.4 - (ALVector) direction** [read, write, assign]

Direction (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.5 - (float) gain** [read, write, assign]

Gain (volume) (OpenAL property).

Reimplemented in [ALChannelSource](#), and [ALSource](#).

**4.9.3.6 - (bool) interruptible** [read, write, assign]

If true, this source may be interrupted when resources are low.

Reimplemented in [ALChannelSource](#), and [ALSource](#).

**4.9.3.7 - (bool) looping** [read, write, assign]

Looping (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.8 - (float) maxDistance** [read, write, assign]

Max distance (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.9 - (float) maxGain** [read, write, assign]

Max gain (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.10 - (float) minGain** [read, write, assign]

Min gain (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.11 - (bool) muted** [read, write, assign]

If true, this source is muted.

Reimplemented in [ALChannelSource](#), and [ALSource](#).

**4.9.3.12 - (float) pan** [read, write, assign]

Pan value (-1.0 = far left, 1.0 = far right).

Note: This effect is simulated by changing the source's X position. Do not use this property if you are modifying the position property as well.

**4.9.3.13 - (bool) paused** [read, write, assign]

If true, this source is currently paused.

Reimplemented in [ALChannelSource](#).

**4.9.3.14 - (float) pitch** [read, write, assign]

Pitch (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.15 - (bool) playing** [read, assign]

If true, this source is currently playing audio.

**4.9.3.16 - (ALPoint) position** [read, write, assign]

Position (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.17 - (float) referenceDistance** [read, write, assign]

Reference distance (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.18 - (float) rolloffFactor** [read, write, assign]

Rolloff factor (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.19 - (int) sourceRelative** [read, write, assign]

Source relative (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.20 - (int) sourceType** [read, assign]

Source type (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.21 - (ALVector) velocity** [read, write, assign]

Velocity (OpenAL property).

Reimplemented in [ALChannelSource](#).

**4.9.3.22 - (float) volume** [read, write, assign]

Volume (alias to gain).

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

- [ALSoundSource.h](#)

## 4.10 ALSoundSourcePool Class Reference

A pool of sound sources, which can be fetched based on availability.

```
#import <ALSoundSourcePool.h>
```

### Public Member Functions

- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (void) - [addSource:](#)  
*Add a source to this pool.*
- (void) - [removeSource:](#)  
*Remove a source from this pool.*
- (id< [ALSoundSource](#) >) - [getFreeSource:](#)  
*Acquire a free or freeable source from this pool.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*
- (void) - [moveToHead:](#)  
*Move a source to the head of the list.*

### Static Public Member Functions

- (id) + [pool](#)  
*Make a new pool.*

### Protected Attributes

- [NSMutableArray](#) \* [sources](#)  
*All sources managed by this pool (id<ALSoundSource>).*

### Properties

- [NSArray](#) \* [sources](#)  
*All sources managed by this pool (id<ALSoundSource>).*

### 4.10.1 Detailed Description

A pool of sound sources, which can be fetched based on availability.

### 4.10.2 Member Function Documentation

#### 4.10.2.1 - (void) addSource: dummy(id<ALSoundSource>) *source*

Add a source to this pool.

##### Parameters

<i>source</i>	The source to add.
---------------	--------------------

#### 4.10.2.2 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

#### 4.10.2.3 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

#### 4.10.2.4 - (id<ALSoundSource>) getFreeSource: dummy(bool) *attemptToInterrupt*

Acquire a free or freeable source from this pool.

It first attempts to find a completely free source. Failing this, it will attempt to interrupt a source and return that (if *attemptToInterrupt* is TRUE).

##### Parameters

<i>attemptToInterrupt</i>	If TRUE, attempt to interrupt sources to free them for use.
---------------------------	---

##### Returns

The freed sound source, or nil if no sources are freeable.

#### 4.10.2.5 - (void) moveToHead: dummy(int) *index*

Move a source to the head of the list.

##### Parameters

<i>index</i>	the index of the source to move.
--------------	----------------------------------

**4.10.2.6 + (id) pool**

Make a new pool.

**Returns**

A new pool.

**4.10.2.7 - (void) removeSource: dummy(id<ALSoundSource>) source**

Remove a source from this pool.

**Parameters**

<i>source</i>	The source to remove.
---------------	-----------------------

**4.10.3 Member Data Documentation****4.10.3.1 - (NSMutableArray\*) sources** [protected]

All sources managed by this pool (id<ALSoundSource>).

**4.10.4 Property Documentation****4.10.4.1 - (NSArray\*) sources** [read, assign]

All sources managed by this pool (id<ALSoundSource>).

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

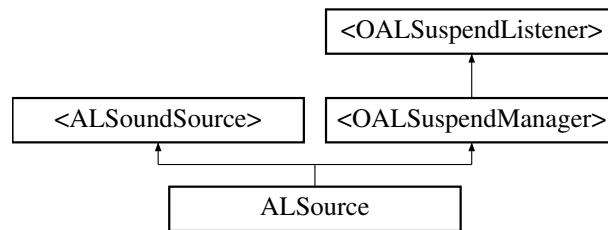
- ALSoundSourcePool.h
- ALSoundSourcePool.m

**4.11 ALSource Class Reference**

A source represents an object that emits sound which can be heard by a listener.

```
#import <ALSource.h>
```

Inheritance diagram for ALSource:



## Public Member Functions

- (id) - [initOnContext](#):  
*Initialize a new source on the specified context.*
- (id< [ALSoundSource](#) >) - [play](#)  
*Play the currently attached buffer.*
- (bool) - [queueBuffer](#):  
*Add a buffer to the buffer queue.*
- (bool) - [queueBuffers](#):  
*Add buffers to the buffer queue.*
- (bool) - [unqueueBuffer](#):  
*Remove a buffer from the buffer queue.*
- (bool) - [unqueueBuffers](#):  
*Remove buffers from the buffer queue.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*
- (void) - [setSuspended](#):  
*(INTERNAL USE) Called by SuspendHandler.*
- (void) - [delayedResumePlayback](#)  
*(INTERNAL USE) Callback for resuming playback after delay to get around OpenAL bug.*

## Static Public Member Functions

- (id) + [source](#)  
*Create a new source.*
- (id) + [sourceOnContext](#):

*Create a new source on the specified context.*

## Protected Attributes

- bool [interruptible](#)  
*If true, this source may be interrupted when resources are low.*
- float [gain](#)  
*Gain (volume) (OpenAL property).*
- bool [muted](#)  
*If true, this source is muted.*
- int [shadowState](#)  
*Shadow value which keeps the correct state value for AL\_PLAYING and AL\_PAUSED.*
- bool [abortPlaybackResume](#)  
*Used to abort a pending playback resume if the user calls stop or pause.*
- [OALAction](#) \* [gainAction](#)  
*Current action operating on the gain control.*
- [OALAction](#) \* [panAction](#)  
*Current action operating on the pan control.*
- [OALAction](#) \* [pitchAction](#)  
*Current action operating on the pitch control.*
- [OALSuspendHandler](#) \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*

## Properties

- [ALBuffer](#) \* [buffer](#)  
*The sound buffer this source is attached to (set to nil to detach the currently attached buffer).*
- int [buffersQueued](#)  
*How many buffers this source has queued.*
- int [buffersProcessed](#)  
*How many of these buffers have been processed during playback.*



- [ALContext \\*](#) [context](#)  
*The context this source was opened on.*
- float [offsetInBytes](#)  
*The offset into the current buffer (in bytes).*
- float [offsetInSamples](#)  
*The offset into the current buffer (in samples).*
- float [offsetInSeconds](#)  
*The offset into the current buffer (in seconds).*
- unsigned int [sourceId](#)  
*OpenAL's ID for this source.*
- int [state](#)  
*The state of this source.*

#### 4.11.1 Detailed Description

A source represents an object that emits sound which can be heard by a listener. This source can have position, velocity, and direction.

#### 4.11.2 Member Function Documentation

##### 4.11.2.1 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

##### 4.11.2.2 - (void) delayedResumePlayback

(INTERNAL USE) Callback for resuming playback after delay to get around OpenAL bug.

##### 4.11.2.3 - (id) initOnContext: dummy(ALContext\*) context

Initialize a new source on the specified context.

##### Parameters

<a href="#">context</a>	the context to create the source on.
-------------------------	--------------------------------------

##### Returns

A new source.

**4.11.2.4 - (id< ALSoundSource >) play**

Play the currently attached buffer.

**Returns**

the source playing the sound, or nil if the sound could not be played.

**4.11.2.5 - (bool) queueBuffer: dummy(ALBuffer\*) buffer**

Add a buffer to the buffer queue.

**Parameters**

<i>buffer</i>	the buffer to add to the queue.
---------------	---------------------------------

**Returns**

TRUE if the operation was successful.

**4.11.2.6 - (bool) queueBuffers: dummy(NSArray\*) buffers**

Add buffers to the buffer queue.

**Parameters**

<i>buffers</i>	the buffers to add to the queue.
----------------	----------------------------------

**Returns**

TRUE if the operation was successful.

**4.11.2.7 - (void) setSuspended: dummy(bool) value**

(INTERNAL USE) Called by SuspendHandler.

**4.11.2.8 + (id) source**

Create a new source.

**Returns**

A new source.

**4.11.2.9 + (id) sourceOnContext: dummy(ALContext\*) context**

Create a new source on the specified context.

**Parameters**

<i>context</i>	the context to create the source on.
----------------	--------------------------------------

**Returns**

A new source.

**4.11.2.10 - (bool) unqueueBuffer: dummy(ALBuffer\*) buffer**

Remove a buffer from the buffer queue.

**Parameters**

<i>buffer</i>	the buffer to remove from the queue.
---------------	--------------------------------------

**Returns**

TRUE if the operation was successful.

**4.11.2.11 - (bool) unqueueBuffers: dummy(NSArray\*) buffers**

Remove buffers from the buffer queue.

**Parameters**

<i>buffers</i>	the buffers to remove from the queue.
----------------	---------------------------------------

**Returns**

TRUE if the operation was successful.

**4.11.3 Member Data Documentation****4.11.3.1 - (bool) abortPlaybackResume** [protected]

Used to abort a pending playback resume if the user calls stop or pause.

**4.11.3.2 - (float) gain** [protected]

Gain (volume) (OpenAL property).

Reimplemented from [<ALSoundSource>](#).

**4.11.3.3 - (OALAction\*) gainAction** [protected]

Current action operating on the gain control.

#### 4.11.3.4 - (bool) interruptible [protected]

If true, this source may be interrupted when resources are low.

Reimplemented from [<ALSoundSource>](#).

#### 4.11.3.5 - (bool) muted [protected]

If true, this source is muted.

Reimplemented from [<ALSoundSource>](#).

#### 4.11.3.6 - (OALAction\*) panAction [protected]

Current action operating on the pan control.

#### 4.11.3.7 - (OALAction\*) pitchAction [protected]

Current action operating on the pitch control.

#### 4.11.3.8 - (int) shadowState [protected]

Shadow value which keeps the correct state value for AL\_PLAYING and AL\_PAUSED.

We need this due to a buggy OpenAL implementation.

#### 4.11.3.9 - (OALSuspendHandler\*) suspendHandler [protected]

Handles suspending and interrupting for this object.

### 4.11.4 Property Documentation

#### 4.11.4.1 - (ALBuffer \*) buffer [read, write, retain]

The sound buffer this source is attached to (set to nil to detach the currently attached buffer).

#### 4.11.4.2 - (int) buffersProcessed [read, assign]

How many of these buffers have been processed during playback.

#### 4.11.4.3 - (int) buffersQueued [read, assign]

How many buffers this source has queued.

**4.11.4.4 - (ALContext \*) context** [read, assign]

The context this source was opened on.

**4.11.4.5 - (float) offsetInBytes** [read, write, assign]

The offset into the current buffer (in bytes).

**4.11.4.6 - (float) offsetInSamples** [read, write, assign]

The offset into the current buffer (in samples).

**4.11.4.7 - (float) offsetInSeconds** [read, write, assign]

The offset into the current buffer (in seconds).

**4.11.4.8 - (unsigned int) sourceId** [read, assign]

OpenAL's ID for this source.

**4.11.4.9 - (int) state** [read, write, assign]

The state of this source.

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

- ALSource.h
- ALSource.m

## 4.12 ALVector Struct Reference

Represents a 3-dimensional vector for certain ObjectAL properties.

```
#include <ALTypes.h>
```

### Public Attributes

- float `x`  
*The "X" coordinate.*
- float `y`  
*The "Y" coordinate.*

- float [z](#)

*The "Z" coordinate.*

#### 4.12.1 Detailed Description

Represents a 3-dimensional vector for certain ObjectAL properties. Properties are the same as for [ALPoint](#).

#### 4.12.2 Member Data Documentation

##### 4.12.2.1 float ALVector::x

The "X" coordinate.

##### 4.12.2.2 float ALVector::y

The "Y" coordinate.

##### 4.12.2.3 float ALVector::z

The "Z" coordinate.

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

- ALTypes.h

### 4.13 ALWrapper Class Reference

A thin wrapper around the C OpenAL API, with a few convenience methods thrown in.

```
#import <ALWrapper.h>
```

#### Public Member Functions

- (BOOL) - [checkIfSuccessful](#)  
*Check the OpenAL error status and log an error message if necessary.*
- (BOOL) - [checkIfSuccessfulWithDevice](#)  
*Check the OpenAL error status and log an error message if necessary.*

### Static Public Member Functions

- (bool) + [genBuffers:numBuffers:](#)  
*Generate buffers.*
- (ALuint) + [genBuffer](#)  
*Generate a buffer.*
- (bool) + [deleteBuffers:numBuffers:](#)  
*Delete buffers.*
- (bool) + [deleteBuffer:](#)  
*Delete a buffer.*
- (bool) + [isBuffer:](#)  
*Check if the speified buffer exists.*
- (bool) + [bufferData:format:data:size:frequency:](#)  
*Load data into a buffer.*
- (bool) + [bufferf:parameter:value:](#)  
*Write a float paramter to a buffer.*
- (bool) + [buffer3f:parameter:v1:v2:v3:](#)  
*Write a 3 float paramter to a buffer.*
- (bool) + [bufferfv:parameter:values:](#)  
*Write a float array paramter to a buffer.*
- (bool) + [bufferi:parameter:value:](#)  
*Write an integer paramter to a buffer.*
- (bool) + [buffer3i:parameter:v1:v2:v3:](#)  
*Write a 3 integer paramter to a buffer.*
- (bool) + [bufferiv:parameter:values:](#)  
*Write an integer array paramter to a buffer.*
- (ALfloat) + [getBufferf:parameter:](#)  
*Read a float paramter from a buffer.*
- (bool) + [getBuffer3f:parameter:v1:v2:v3:](#)  
*Read a 3 float paramter from a buffer.*
- (bool) + [getBufferfv:parameter:values:](#)  
*Read a float array paramter from a buffer.*

- (ALint) + [getBufferi:parameter:](#)  
*Read an integer paramter from a buffer.*
- (bool) + [getBuffer3i:parameter:v1:v2:v3:](#)  
*Read a 3 integer paramter from a buffer.*
- (bool) + [getBufferiv:parameter:values:](#)  
*Read an integer array paramter from a buffer.*
- (bool) + [genSources:numSources:](#)  
*Generate sources.*
- (ALuint) + [genSource](#)  
*Generate a source.*
- (bool) + [deleteSources:numSources:](#)  
*Delete sources.*
- (bool) + [deleteSource:](#)  
*Delete a source.*
- (bool) + [isSource:](#)  
*Check if the speified source exists.*
- (bool) + [sourcePlay:](#)  
*Play a source.*
- (bool) + [sourcePlayv:numSources:](#)  
*Play a bunch of sources.*
- (bool) + [sourcePause:](#)  
*Pause a source.*
- (bool) + [sourcePausev:numSources:](#)  
*Pause a bunch of sources.*
- (bool) + [sourceStop:](#)  
*Stop a source.*
- (bool) + [sourceStopv:numSources:](#)  
*Stop a bunch of sources.*
- (bool) + [sourceRewind:](#)  
*Rewind a source.*



- (bool) + [sourceRewindv:numSources:](#)  
*Rewind a bunch of sources.*
- (bool) + [sourceQueueBuffers:numBuffers:bufferIds:](#)  
*Queue buffers into a source for sequential playback.*
- (bool) + [sourceUnqueueBuffers:numBuffers:bufferIds:](#)  
*Unqueue previously queued buffers.*
- (bool) + [sourcef:parameter:value:](#)  
*Write a float paramter to a source.*
- (bool) + [source3f:parameter:v1:v2:v3:](#)  
*Write a 3 float paramter to a source.*
- (bool) + [sourcefv:parameter:values:](#)  
*Write a float array paramter to a source.*
- (bool) + [sourcei:parameter:value:](#)  
*Write an integer paramter to a source.*
- (bool) + [source3i:parameter:v1:v2:v3:](#)  
*Write a 3 integer paramter to a source.*
- (bool) + [sourceiv:parameter:values:](#)  
*Write an integer array paramter to a source.*
- (ALfloat) + [getSourcef:parameter:](#)  
*Read a float paramter from a source.*
- (bool) + [getSource3f:parameter:v1:v2:v3:](#)  
*Read a 3 float paramter from a source.*
- (bool) + [getSourcefv:parameter:values:](#)  
*Read a float array paramter from a source.*
- (ALint) + [getSourcei:parameter:](#)  
*Read an integer paramter from a source.*
- (bool) + [getSource3i:parameter:v1:v2:v3:](#)  
*Read a 3 integer paramter from a source.*
- (bool) + [getSourceiv:parameter:values:](#)  
*Read an integer array paramter from a source.*
- (bool) + [listenerf:value:](#)

*Write a float paramter to the current listener.*

- (bool) + [listener3f:v1:v2:v3:](#)

*Write a 3 float paramter to the current listener.*

- (bool) + [listenerfv:values:](#)

*Write a float array paramter to the current listener.*

- (bool) + [listeneri:value:](#)

*Write an integer paramter to the current listener.*

- (bool) + [listener3i:v1:v2:v3:](#)

*Write a 3 integer paramter to the current listener.*

- (bool) + [listeneriv:values:](#)

*Write an integer array paramter to the current listener.*

- (ALfloat) + [getListenerf:](#)

*Read a float paramter from the current listener.*

- (bool) + [getListener3f:v1:v2:v3:](#)

*Read a 3 float paramter from the current listener.*

- (bool) + [getListenerfv:values:](#)

*Read a float array paramter from the current listener.*

- (ALint) + [getListeneri:](#)

*Read an integer paramter from the current listener.*

- (bool) + [getListener3i:v1:v2:v3:](#)

*Read a 3 integer paramter from the current listener.*

- (bool) + [getListeneriv:values:](#)

*Read an integer array paramter from the current listener.*

- (bool) + [enable:](#)

*Enable a capability.*

- (bool) + [disable:](#)

*Disable a capability.*

- (bool) + [isEnabled:](#)

*Check if a capability is enabled.*

- (bool) + [getBoolean:](#)

*Get a boolean parameter.*

- (ALdouble) + [getDouble](#):  
*Get a double parameter.*
- (ALfloat) + [getFloat](#):  
*Get a float parameter.*
- (ALint) + [getInteger](#):  
*Get an integer parameter.*
- (NSString \*) + [getString](#):  
*Get a string parameter.*
- (NSArray \*) + [getNullSeparatedStringList](#):  
*Get a string list parameter.*
- (NSArray \*) + [getSpaceSeparatedStringList](#):  
*Get a string list parameter.*
- (bool) + [getBooleanv:values](#):  
*Get a boolean array parameter.*
- (bool) + [getDoublev:values](#):  
*Get a double array parameter.*
- (bool) + [getFloatv:values](#):  
*Get a float array parameter.*
- (bool) + [getIntegerv:values](#):  
*Get an integer array parameter.*
- (bool) + [distanceModel](#):  
*Set the distance model.*
- (bool) + [dopplerFactor](#):  
*Set the doppler factor.*
- (bool) + [speedOfSound](#):  
*Set the speed of sound.*
- (bool) + [isExtensionPresent](#):  
*Check if an extension is present.*
- (void \*) + [getProcAddress](#):  
*Get the address of a procedure.*

- (ALenum) + [getEnumValue](#):  
*Get the enum value from its name.*
- (ALCdevice \*) + [openDevice](#):  
*Open a device.*
- (bool) + [closeDevice](#):  
*Close a device.*
- (ALCcontext \*) + [createContext:attributes](#):  
*Create an OpenAL context.*
- (bool) + [makeContextCurrent](#):  
*Make the specified context the current context.*
- (bool) + [makeContextCurrent:deviceReference](#):  
*Make the specified context the current context, passing in a device reference for more informative logging info.*
- (void) + [processContext](#):  
*Process a context.*
- (void) + [suspendContext](#):  
*Suspend a context.*
- (void) + [destroyContext](#):  
*Destroy a context.*
- (ALCcontext \*) + [getCurrentContext](#):  
*Get the current context.*
- (ALCdevice \*) + [getContextsDevice](#):  
*Get the device a context was created from.*
- (ALCdevice \*) + [getContextsDevice:deviceReference](#):  
*Get the device a context was created from, passing in a device reference for more informative logging info.*
- (bool) + [isExtensionPresent:name](#):  
*Check if an extension is present on a device.*
- (void \*) + [getProcAddress:name](#):  
*Get the address of a procedure for a device.*
- (ALenum) + [getEnumValue:name](#):  
*Get the enum value from its name.*

- (NSString \*) + [getString:attribute:](#)  
*Get a string attribute.*
- (NSArray \*) + [getNullSeparatedStringList:attribute:](#)  
*Get a string list attribute.*
- (NSArray \*) + [getSpaceSeparatedStringList:attribute:](#)  
*Get a string list attribute.*
- (ALint) + [getInteger:attribute:](#)  
*Get an integer attribute.*
- (bool) + [getIntegerv:attribute:size:data:](#)  
*Get an integer array attribute.*
- (ALCdevice \*) + [openCaptureDevice:frequency:format:bufferSize:](#)  
*\*UNSUPPORTED ON IOS\* Open an audio capture device.*
- (bool) + [closeCaptureDevice:](#)  
*Close a capture device.*
- (bool) + [startCapture:](#)  
*Start capturing audio data.*
- (bool) + [stopCapture:](#)  
*Stop capturing audio data.*
- (bool) + [captureSamples:buffer:numSamples:](#)  
*Get captured samples from a device.*
- (ALdouble) + [getMixerOutputDataRate](#)  
*Get the iOS device's mixer output data rate.*
- (void) + [setMixerOutputDataRate:](#)  
*Set the iOS device's mixer output data rate.*
- (bool) + [bufferDataStatic:format:data:size:frequency:](#)  
*Load data into a buffer.*
- (NSArray \*) + [decodeNullSeparatedStringList:](#)  
*Decode an OpenAL supplied NULL-separated string list into an NSArray.*
- (NSArray \*) + [decodeSpaceSeparatedStringList:](#)  
*Decode an OpenAL supplied space-separated string list into an NSArray.*

### 4.13.1 Detailed Description

A thin wrapper around the C OpenAL API, with a few convenience methods thrown in. Wherever possible, methods return the requested data rather than requiring a pointer to be passed in. Besides collecting the API calls into a single global object, all calls are combined with an error check. Any OpenAL errors that occur will be logged if error logging is enabled.

### 4.13.2 Member Function Documentation

**4.13.2.1** `+(bool) buffer3f: dummy(ALuint) bufferId parameter:(ALenum) parameter v1:(ALfloat) v1 v2:(ALfloat) v2 v3:(ALfloat) v3`

Write a 3 float paramter to a buffer.

#### Parameters

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	the parameter to write to.
<i>v1</i>	The first value to write.
<i>v2</i>	The second value to write.
<i>v3</i>	The third value to write.

#### Returns

TRUE if the operation was successful.

**4.13.2.2** `+(bool) buffer3i: dummy(ALuint) bufferId parameter:(ALenum) parameter v1:(ALint) v1 v2:(ALint) v2 v3:(ALint) v3`

Write a 3 integer paramter to a buffer.

#### Parameters

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to write to.
<i>v1</i>	The first value to write.
<i>v2</i>	The second value to write.
<i>v3</i>	The third value to write.

#### Returns

TRUE if the operation was successful.

**4.13.2.3** `+(bool) bufferData: dummy(ALuint) bufferId format:(ALenum) format data:(const ALvoid*) data size:(ALsizei) size frequency:(ALsizei) frequency`

Load data into a buffer.

**Parameters**

<i>bufferId</i>	The ID of the buffer to load data into.
<i>format</i>	The format of the data being loaded (typically AL_FORMAT_MONO16 or AL_FORMAT_STEREO16).
<i>data</i>	The audio data.
<i>size</i>	The size of the data in bytes.
<i>frequency</i>	The sample frequency of the data.

#### 4.13.2.4 + (bool) **bufferDataStatic:** **dummy**(ALuint) *bufferId* **format:**(ALenum) *format* **data:**(const ALvoid\*) *data* **size:**(ALsizei) *size* **frequency:**(ALsizei) *frequency*

Load data into a buffer.

Unlike "bufferData", with this method the buffer will use the passed in data buffer directly rather than allocating its own memory and copying from the data buffer.

**Parameters**

<i>bufferId</i>	The ID of the buffer to load data into.
<i>format</i>	The format of the data being loaded (typically AL_FORMAT_MONO16 or AL_FORMAT_STEREO16).
<i>data</i>	The audio data.
<i>size</i>	The size of the data in bytes.
<i>frequency</i>	The sample frequency of the data.

#### 4.13.2.5 + (bool) **bufferf:** **dummy**(ALuint) *bufferId* **parameter:**(ALenum) *parameter* **value:**(ALfloat) *value*

Write a float paramter to a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to write to.
<i>value</i>	The value to write.

**Returns**

TRUE if the operation was successful.

#### 4.13.2.6 + (bool) **bufferfv:** **dummy**(ALuint) *bufferId* **parameter:**(ALenum) *parameter* **values:**(ALfloat\*) *values*

Write a float array paramter to a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to write to.
<i>values</i>	The values to write.

**Returns**

TRUE if the operation was successful.

#### 4.13.2.7 + (bool) *bufferId*: dummy(ALuint) *bufferId* parameter:(ALenum) *parameter* value:(ALint) *value*

Write an integer paramter to a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to write to.
<i>value</i>	The value to write.

**Returns**

TRUE if the operation was successful.

#### 4.13.2.8 + (bool) *bufferId*: dummy(ALuint) *bufferId* parameter:(ALenum) *parameter* values:(ALint\*) *values*

Write an integer array paramter to a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to write to.
<i>values</i>	The values to write.

**Returns**

TRUE if the operation was successful.

#### 4.13.2.9 + (bool) *captureSamples*: dummy(ALCdevice\*) *device* buffer:(ALCvoid\*) *buffer* numSamples:(ALCsizei) *numSamples*

Get captured samples from a device.

**Parameters**

<i>device</i>	the device to fetch samples from.
<i>buffer</i>	the buffer to copy the samples into.



<i>numSamples</i>	the number of samples to fetch.
-------------------	---------------------------------

#### 4.13.2.10 - (BOOL) checkIfSuccessful dummy(const char \*) *contextInfo*

Check the OpenAL error status and log an error message if necessary.

##### Parameters

<i>contextInfo</i>	Contextual information to add when logging an error.
--------------------	--

##### Returns

TRUE if the operation was successful (no error).

#### 4.13.2.11 - (BOOL) checkIfSuccessfulWithDevice dummy(const char \*) *contextInfo* (ALCdevice \*) *device*

Check the OpenAL error status and log an error message if necessary.

##### Parameters

<i>contextInfo</i>	Contextual information to add when logging an error.
<i>device</i>	The device to check for errors on.

##### Returns

TRUE if the operation was successful (no error).

#### 4.13.2.12 + (bool) closeCaptureDevice: dummy(ALCdevice\*) *device*

Close a capture device.

##### Parameters

<i>device</i>	The device to close.
---------------	----------------------

##### Returns

TRUE if the operation was successful.

#### 4.13.2.13 + (bool) closeDevice: dummy(ALCdevice\*) *device*

Close a device.

**Parameters**

<i>device</i>	The device to close.
---------------	----------------------

**Returns**

TRUE if the operation was successful.

#### 4.13.2.14 + (ALCcontext \*) createContext: dummy(ALCdevice\*) *device* attributes:(ALCint\*) *attributes*

Create an OpenAL context.

**Parameters**

<i>device</i>	The device to open the context on.
<i>attributes</i>	The attributes to use when creating the context.

**Returns**

The new context.

#### 4.13.2.15 + (NSArray\*) decodeNullSeparatedStringList: dummy(const ALCchar \*) *source*

Decode an OpenAL supplied NULL-separated string list into an NSArray.

**Parameters**

<i>source</i>	the string list as supplied by OpenAL.
---------------	--

**Returns**

the string list in an NSArray of NSString.

#### 4.13.2.16 + (NSArray\*) decodeSpaceSeparatedStringList: dummy(const ALCchar \*) *source*

Decode an OpenAL supplied space-separated string list into an NSArray.

**Parameters**

<i>source</i>	the string list as supplied by OpenAL.
---------------	--

**Returns**

the string list in an NSArray of NSString.

**4.13.2.17 + (bool) deleteBuffer: dummy(ALuint) *bufferId***

Delete a buffer.

**Parameters**

<i>bufferId</i>	The ID of the buffer to delete.
-----------------	---------------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.18 + (bool) deleteBuffers: dummy(ALuint\*) *bufferIds* numBuffers:(ALsizei) *numBuffers***

Delete buffers.

**Parameters**

<i>bufferIds</i>	Pointer to an array containing the buffer IDs.
<i>numBuffers</i>	the number of buffers to delete.

**Returns**

TRUE if the operation was successful.

**4.13.2.19 + (bool) deleteSource: dummy(ALuint) *sourceId***

Delete a source.

**Parameters**

<i>sourceId</i>	The ID of the source to delete.
-----------------	---------------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.20 + (bool) deleteSources: dummy(ALuint\*) *sourceIds* numSources:(ALsizei) *numSources***

Delete sources.

**Parameters**

<i>sourceIds</i>	Pointer to an array containing the source IDs.
<i>numSources</i>	the number of sources to delete.

**Returns**

TRUE if the operation was successful.

**4.13.2.21 + (void) destroyContext: dummy(ALCcontext\*) context**

Destroy a context.

**Parameters**

<i>context</i>	The context to destroy.
----------------	-------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.22 + (bool) disable: dummy(ALenum) capability**

Disable a capability.

**Parameters**

<i>capability</i>	The capability to disable.
-------------------	----------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.23 + (bool) distanceModel: dummy(ALenum) value**

Set the distance model.

**Parameters**

<i>value</i>	The value to set.
--------------	-------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.24 + (bool) dopplerFactor: dummy(ALfloat) value**

Set the doppler factor.

**Parameters**

<i>value</i>	The value to set.
--------------	-------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.25 + (bool) enable: dummy(ALenum) capability**

Enable a capability.

**Parameters**

<i>capability</i>	The capability to enable.
-------------------	---------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.26 + (ALuint) genBuffer**

Generate a buffer.

**Returns**

the buffer's ID.

**4.13.2.27 + (bool) genBuffers: dummy(ALuint\*) bufferIds numBuffers:(ALsizei) numBuffers**

Generate buffers.

**Parameters**

<i>bufferIds</i>	Pointer to an array that will receive the buffer IDs.
<i>numBuffers</i>	the number of buffers to generate.

**Returns**

TRUE if the operation was successful.

**4.13.2.28 + (ALuint) genSource**

Generate a source.

**Returns**

the source's ID.

#### 4.13.2.29 + (bool) genSources: dummy(ALuint\*) *sourceIds* numSources:(ALsizei) *numSources*

Generate sources.

##### Parameters

<i>sourceIds</i>	Pointer to an array that will receive the source IDs.
<i>numSources</i>	the number of sources to generate.

##### Returns

TRUE if the operation was successful.

#### 4.13.2.30 + (bool) getBoolean: dummy(ALenum) *parameter*

Get a boolean parameter.

##### Parameters

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

##### Returns

The parameter's current value.

#### 4.13.2.31 + (bool) getBooleanv: dummy(ALenum) *parameter* values:(ALboolean\*) *values*

Get a boolean array parameter.

##### Parameters

<i>parameter</i>	The parameter to fetch.
<i>values</i>	An array to hold the result.

##### Returns

TRUE if the operation was successful.

#### 4.13.2.32 + (bool) getBuffer3f: dummy(ALuint) *bufferId* parameter:(ALenum) *parameter* v1:(ALfloat\*) *v1* v2:(ALfloat\*) *v2* v3:(ALfloat\*) *v3*

Read a 3 float paramter from a buffer.

##### Parameters

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to read.

<i>v1</i>	The first value to read.
<i>v2</i>	The second value to read.
<i>v3</i>	The third value to read.

**Returns**

TRUE if the operation was successful.

**4.13.2.33** + (bool) **getBuffer3i:** dummy(ALuint) *bufferId* parameter:(ALenum) *parameter*  
*v1*:(ALint\*) *v1* *v2*:(ALint\*) *v2* *v3*:(ALint\*) *v3*

Read a 3 integer paramter from a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to read.
<i>v1</i>	The first value to read.
<i>v2</i>	The second value to read.
<i>v3</i>	The third value to read.

**Returns**

TRUE if the operation was successful.

**4.13.2.34** + (ALfloat) **getBufferf:** dummy(ALuint) *bufferId* parameter:(ALenum) *parameter*

Read a float paramter from a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to read.

**Returns**

The parameter's value.

**4.13.2.35** + (bool) **getBufferfv:** dummy(ALuint) *bufferId* parameter:(ALenum) *parameter*  
*values*:(ALfloat\*) *values*

Read a float array paramter from a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to read.
<i>values</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

**4.13.2.36 + (ALint) getBufferi: dummy(ALuint) *bufferId* parameter:(ALenum) *parameter***

Read an integer paramter from a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to read.

**Returns**

The parameter's value.

**4.13.2.37 + (bool) getBufferiv: dummy(ALuint) *bufferId* parameter:(ALenum) *parameter* values:(ALint\*) *values***

Read an integer array paramter from a buffer.

**Parameters**

<i>bufferId</i>	The buffer's ID.
<i>parameter</i>	The parameter to read.
<i>values</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

**4.13.2.38 + (ALCdevice \*) getContextsDevice: dummy(ALCcontext\*) *context***

Get the device a context was created from.

**Parameters**

<i>context</i>	The context.
----------------	--------------

**Returns**

The context's device.



**4.13.2.39 + (ALCdevice \*) getContextsDevice: dummy(ALCcontext\*) context deviceReference:(ALCdevice\*) deviceReference**

Get the device a context was created from, passing in a device reference for more informative logging info.

**Parameters**

<i>context</i>	The context.
<i>deviceReference</i>	The device reference to use when logging an error.

**Returns**

The context's device.

**4.13.2.40 + (ALCcontext \*) getCurrentContext**

Get the current context.

**Returns**

the current context.

**4.13.2.41 + (ALdouble) getDouble: dummy(ALenum) parameter**

Get a double parameter.

**Parameters**

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

**Returns**

The parameter's current value.

**4.13.2.42 + (bool) getDoublev: dummy(ALenum) parameter values:(ALdouble\*) values**

Get a double array parameter.

**Parameters**

<i>parameter</i>	The parameter to fetch.
<i>values</i>	An array to hold the result.

**Returns**

TRUE if the operation was successful.

**4.13.2.43 + (ALenum) getEnumValue: dummy(NSString\*) enumName**

Get the enum value from its name.

**Parameters**

<i>enumName</i>	the name of the enum value.
-----------------	-----------------------------

**Returns**

The enum value.

**4.13.2.44 + (ALenum) getEnumValue: dummy(ALCdevice\*) device name:(NSString\*) enumName**

Get the enum value from its name.

**Parameters**

<i>device</i>	The device to check on.
<i>enumName</i>	the name of the enum value.

**Returns**

The enum value.

**4.13.2.45 + (ALfloat) getFloat: dummy(ALenum) parameter**

Get a float parameter.

**Parameters**

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

**Returns**

The parameter's current value.

**4.13.2.46 + (bool) getFloatv: dummy(ALenum) parameter values:(ALfloat\*) values**

Get a float array parameter.

**Parameters**

<i>parameter</i>	The parameter to fetch.
<i>values</i>	An array to hold the result.

**Returns**

TRUE if the operation was successful.

**4.13.2.47 + (ALint) getInteger: dummy(ALenum) *parameter***

Get an integer parameter.

**Parameters**

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

**Returns**

The parameter's current value.

**4.13.2.48 + (ALint) getInteger: dummy(ALCdevice\*) *device* attribute:(ALenum) *attribute***

Get an integer attribute.

**Parameters**

<i>device</i>	The device to read the attribute from.
<i>attribute</i>	The attribute to fetch.

**Returns**

The parameter's current value.

**4.13.2.49 + (bool) getIntegerv: dummy(ALCdevice\*) *device* attribute:(ALenum) *attribute* size:(ALsizei) *size* data:(ALCint\*) *data***

Get an integer array attribute.

**Parameters**

<i>device</i>	The device to read the attribute from.
<i>attribute</i>	The attribute to read.
<i>size</i>	the size of the receiving array.
<i>data</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

#### 4.13.2.50 + (bool) getIntegerv: dummy(ALenum) parameter values:(ALint\*) values

Get an integer array parameter.

##### Parameters

<i>parameter</i>	The parameter to fetch.
<i>values</i>	An array to hold the result.

##### Returns

TRUE if the operation was successful.

#### 4.13.2.51 + (bool) getListener3f: dummy(ALenum) parameter v1:(ALfloat\*) v1 v2:(ALfloat\*) v2 v3:(ALfloat\*) v3

Read a 3 float paramter from the current listener.

##### Parameters

<i>parameter</i>	The parameter to read.
<i>v1</i>	The first value to read.
<i>v2</i>	The second value to read.
<i>v3</i>	The third value to read.

##### Returns

TRUE if the operation was successful.

#### 4.13.2.52 + (bool) getListener3i: dummy(ALenum) parameter v1:(ALint\*) v1 v2:(ALint\*) v2 v3:(ALint\*) v3

Read a 3 integer paramter from the current listener.

##### Parameters

<i>parameter</i>	The parameter to read.
<i>v1</i>	The first value to read.
<i>v2</i>	The second value to read.
<i>v3</i>	The third value to read.

##### Returns

TRUE if the operation was successful.

#### 4.13.2.53 + (ALfloat) getListenerf: dummy(ALenum) parameter

Read a float paramter from the current listener.

**Parameters**

<i>parameter</i>	The parameter to read.
------------------	------------------------

**Returns**

The parameter's value.

**4.13.2.54 + (bool) getListenerfv: dummy(ALenum) *parameter* values:(ALfloat\*) *values***

Read a float array paramter from the current listener.

**Parameters**

<i>parameter</i>	The parameter to read.
<i>values</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

**4.13.2.55 + (ALint) getListeneri: dummy(ALenum) *parameter***

Read an integer paramter from the current listener.

**Parameters**

<i>parameter</i>	The parameter to read.
------------------	------------------------

**Returns**

The parameter's value.

**4.13.2.56 + (bool) getListeneriv: dummy(ALenum) *parameter* values:(ALint\*) *values***

Read an integer array paramter from the current listener.

**Parameters**

<i>parameter</i>	The parameter to read.
<i>values</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

**4.13.2.57 + (ALdouble) getMixerOutputDataRate**

Get the iOS device's mixer output data rate.

**Returns**

The mixer output data rate.

**4.13.2.58 + (NSArray \*) getNullSeparatedStringList: dummy(ALenum) parameter**

Get a string list parameter.

Use this method for OpenAL parameters that return a null separated list.

**Parameters**

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

**Returns**

The parameter's current value (as an array of NSString\*).

**4.13.2.59 + (NSArray \*) getNullSeparatedStringList: dummy(ALCdevice\*) device attribute:(ALenum) attribute**

Get a string list attribute.

Use this method for OpenAL attributes that return a null separated list.

**Parameters**

<i>device</i>	The device to read the attribute from.
<i>attribute</i>	The attribute to fetch.

**Returns**

The parameter's current value (as an array of NSString\*).

**4.13.2.60 + (void \*) getProcAddress: dummy(NSString\*) functionName**

Get the address of a procedure.

**Parameters**

<i>function-Name</i>	The name of the procedure to fetch.
----------------------	-------------------------------------

**Returns**

A pointer to the procedure, or NULL if it wasn't found.

**4.13.2.61** + (void \*) **getProcAddress:** dummy(ALCdevice\*) *device name:(NSString\*)  
functionName*

Get the address of a procedure for a device.

**Parameters**

<i>device</i>	The device to check on.
<i>function- Name</i>	The name of the procedure to check for.

**Returns**

The procedure's address, or NULL if not found.

**4.13.2.62** + (bool) **getSource3f:** dummy(ALuint) *sourceId parameter:(ALenum) parameter  
v1:(ALfloat\*) v1 v2:(ALfloat\*) v2 v3:(ALfloat\*) v3*

Read a 3 float paramter from a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to read.
<i>v1</i>	The first value to read.
<i>v2</i>	The second value to read.
<i>v3</i>	The third value to read.

**Returns**

TRUE if the operation was successful.

**4.13.2.63** + (bool) **getSource3i:** dummy(ALuint) *sourceId parameter:(ALenum) parameter  
v1:(ALint\*) v1 v2:(ALint\*) v2 v3:(ALint\*) v3*

Read a 3 integer paramter from a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to read.
<i>v1</i>	The first value to read.
<i>v2</i>	The second value to read.
<i>v3</i>	The third value to read.

**Returns**

TRUE if the operation was successful.

**4.13.2.64 + (ALfloat) getSourcef: dummy(ALuint) sourceId parameter:(ALenum) parameter**

Read a float paramter from a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to read.

**Returns**

The parameter's value.

**4.13.2.65 + (bool) getSourcefv: dummy(ALuint) sourceId parameter:(ALenum) parameter values:(ALfloat\*) values**

Read a float array paramter from a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to read.
<i>values</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

**4.13.2.66 + (ALint) getSourcei: dummy(ALuint) sourceId parameter:(ALenum) parameter**

Read an integer paramter from a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to read.

**Returns**

The parameter's value.



**4.13.2.67 + (bool) getSourceiv: dummy(ALuint) sourceId parameter:(ALenum) parameter values:(ALint\*) values**

Read an integer array paramter from a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to read.
<i>values</i>	An array to store the values.

**Returns**

TRUE if the operation was successful.

**4.13.2.68 + (NSArray \*) getSpaceSeparatedStringList: dummy(ALenum) parameter**

Get a string list parameter.

Use this method for OpenAL parameters that return a space separated list.

**Parameters**

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

**Returns**

The parameter's current value (as an array of NSString\*).

**4.13.2.69 + (NSArray \*) getSpaceSeparatedStringList: dummy(ALCdevice\*) device attribute:(ALenum) attribute**

Get a string list attribute.

Use this method for OpenAL attributes that return a space separated list.

**Parameters**

<i>device</i>	The device to read the attribute from.
<i>attribute</i>	The attribute to fetch.

**Returns**

The parameter's current value (as an array of NSString\*).

**4.13.2.70 + (NSString \*) getString: dummy(ALenum) parameter**

Get a string parameter.

**Parameters**

<i>parameter</i>	The parameter to fetch.
------------------	-------------------------

**Returns**

The parameter's current value.

**4.13.2.71 + (NSString \*) getString: dummy(ALCdevice\*) device attribute:(ALenum) attribute**

Get a string attribute.

**Parameters**

<i>device</i>	The device to read the attribute from.
<i>attribute</i>	The attribute to fetch.

**Returns**

The parameter's current value.

**4.13.2.72 + (bool) isBuffer: dummy(ALuint) bufferId**

Check if the speified buffer exists.

**Parameters**

<i>bufferId</i>	The ID of the buffer to query.
-----------------	--------------------------------

**Returns**

TRUE if the buffer exists.

**4.13.2.73 + (bool) isEnabled: dummy(ALenum) capability**

Check if a capability is enabled.

**Parameters**

<i>capability</i>	The capability to check.
-------------------	--------------------------

**Returns**

TRUE if the capability is enabled.

**4.13.2.74 + (bool) isExtensionPresent: dummy(NSString\*) extensionName**

Check if an extension is present.

**Parameters**

<i>extension-Name</i>	The name of the extension to check.
-----------------------	-------------------------------------

**Returns**

TRUE if the extension is present.

#### 4.13.2.75 + (bool) isExtensionPresent: dummy(ALCdevice\*) device name:(NSString\*) extensionName

Check if an extension is present on a device.

**Parameters**

<i>device</i>	The device to check for an extension on.
<i>extension-Name</i>	The name of the extension to check for.

**Returns**

TRUE if the extension is present.

#### 4.13.2.76 + (bool) isSource: dummy(ALuint) sourceId

Check if the speified source exists.

**Parameters**

<i>sourceId</i>	The ID of the source to query.
-----------------	--------------------------------

**Returns**

TRUE if the buffer exists.

#### 4.13.2.77 + (bool) listener3f: dummy(ALenum) parameter v1:(ALfloat) v1 v2:(ALfloat) v2 v3:(ALfloat) v3

Write a 3 float paramter to the current listener.

**Parameters**

<i>parameter</i>	the parameter to write to.
<i>v1</i>	The first value to write.
<i>v2</i>	The second value to write.
<i>v3</i>	The third value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.78** + (bool) listener3i: dummy(ALenum) *parameter* v1:(ALint) v1 v2:(ALint) v2 v3:(ALint) v3

Write a 3 integer paramter to the current listener.

**Parameters**

<i>parameter</i>	The parameter to write to.
<i>v1</i>	The first value to write.
<i>v2</i>	The second value to write.
<i>v3</i>	The third value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.79** + (bool) listenerf: dummy(ALenum) *parameter* value:(ALfloat) *value*

Write a float paramter to the current listener.

**Parameters**

<i>parameter</i>	The parameter to write to.
<i>value</i>	The value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.80** + (bool) listenerfv: dummy(ALenum) *parameter* values:(ALfloat\*) *values*

Write a float array paramter to the current listener.

**Parameters**

<i>parameter</i>	The parameter to write to.
<i>values</i>	The values to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.81 + (bool) listeneri: dummy(ALenum) parameter value:(ALint) value**

Write an integer paramter to the current listener.

**Parameters**

<i>parameter</i>	The parameter to write to.
<i>value</i>	The value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.82 + (bool) listeneriv: dummy(ALenum) parameter values:(ALint\*) values**

Write an integer array paramter to the current listener.

**Parameters**

<i>parameter</i>	The parameter to write to.
<i>values</i>	The values to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.83 + (bool) makeContextCurrent: dummy(ALCcontext\*) context**

Make the specified context the current context.

**Parameters**

<i>context</i>	the context to make current.
----------------	------------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.84 + (bool) makeContextCurrent: dummy(ALCcontext\*) context  
deviceReference:(ALCdevice\*) deviceReference**

Make the specified context the current context, passing in a device reference for more informative logging info.

**Parameters**

<i>context</i>	The context to make current.
<i>deviceReference</i>	The device reference to use when logging an error.

**Returns**

TRUE if the operation was successful.

**4.13.2.85** + (ALCdevice \*) openCaptureDevice: dummy(NSString\*) *deviceName*  
frequency:(ALCuint) *frequency* format:(ALCenum) *format* bufferSize:(ALCsizei)  
*bufferSize*

\*UNSUPPORTED ON IOS\* Open an audio capture device.

**Parameters**

<i>deviceName</i>	The name of the device to open (nil = open the default device).
<i>frequency</i>	The sampling frequency to use.
<i>format</i>	The format to capture the data as.
<i>bufferSize</i>	The size of capture buffer to use.

**Returns**

The opened device, or nil if an error occurred.

**4.13.2.86** + (ALCdevice \*) openDevice: dummy(NSString\*) *deviceName*

Open a device.

**Parameters**

<i>deviceName</i>	The name of the device to open (nil = open the default device).
-------------------	---

**Returns**

The opened device, or nil on failure.

**4.13.2.87** + (void) processContext: dummy(ALCcontext\*) *context*

Process a context.

**Parameters**

<i>context</i>	The context to process.
----------------	-------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.88** + (void) setMixerOutputDataRate: dummy(ALdouble) *frequency*

Set the iOS device's mixer output data rate.

**Parameters**

<i>frequency</i>	The output data rate (frequency).
------------------	-----------------------------------

**4.13.2.89** + (bool) source3f: dummy(ALuint) *sourceId* parameter:(ALenum) *parameter*  
v1:(ALfloat) v1 v2:(ALfloat) v2 v3:(ALfloat) v3

Write a 3 float paramter to a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	the parameter to write to.
<i>v1</i>	The first value to write.
<i>v2</i>	The second value to write.
<i>v3</i>	The third value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.90** + (bool) source3i: dummy(ALuint) *sourceId* parameter:(ALenum) *parameter*  
v1:(ALint) v1 v2:(ALint) v2 v3:(ALint) v3

Write a 3 integer paramter to a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to write to.
<i>v1</i>	The first value to write.
<i>v2</i>	The second value to write.
<i>v3</i>	The third value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.91** + (bool) sourcef: dummy(ALuint) *sourceId* parameter:(ALenum) *parameter*  
value:(ALfloat) *value*

Write a float paramter to a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to write to.
<i>value</i>	The value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.92** + (bool) sourcefv: dummy(ALuint) *sourceId* parameter:(ALenum) *parameter*  
values:(ALfloat\*) *values*

Write a float array paramter to a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to write to.
<i>values</i>	The values to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.93** + (bool) sourcei: dummy(ALuint) *sourceId* parameter:(ALenum) *parameter*  
value:(ALint) *value*

Write an integer paramter to a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to write to.
<i>value</i>	The value to write.

**Returns**

TRUE if the operation was successful.

**4.13.2.94** + (bool) sourceiv: dummy(ALuint) *sourceId* parameter:(ALenum) *parameter*  
values:(ALint\*) *values*

Write an integer array paramter to a source.

**Parameters**

<i>sourceId</i>	The source's ID.
<i>parameter</i>	The parameter to write to.
<i>values</i>	The values to write.

**Returns**

TRUE if the operation was successful.



**4.13.2.95 + (bool) sourcePause: dummy(ALuint) *sourceId***

Pause a source.

**Parameters**

<i>sourceId</i>	The ID of the source to pause.
-----------------	--------------------------------

**Returns**

TRUE if the operation is successful.

**4.13.2.96 + (bool) sourcePausev: dummy(ALuint\*) *sourceIds* numSources:(ALsizei) *numSources***

Pause a bunch of sources.

**Parameters**

<i>sourceIds</i>	The sources to pause.
<i>numSources</i>	The number of sources in <i>sourceIds</i> .

**Returns**

TRUE if the operation is successful.

**4.13.2.97 + (bool) sourcePlay: dummy(ALuint) *sourceId***

Play a source.

**Parameters**

<i>sourceId</i>	The ID of the source to play.
-----------------	-------------------------------

**Returns**

TRUE if the buffer exists.

**4.13.2.98 + (bool) sourcePlayv: dummy(ALuint\*) *sourceIds* numSources:(ALsizei) *numSources***

Play a bunch of sources.

**Parameters**

<i>sourceIds</i>	The sources to play.
<i>numSources</i>	The number of sources in <i>sourceIds</i> .

**Returns**

TRUE if the operation is successful.

**4.13.2.99** + (bool) sourceQueueBuffers: dummy(ALuint) *sourceId* numBuffers:(ALsizei)  
*numBuffers* bufferIds:(ALuint\*) *bufferIds*

Queue buffers into a source for sequential playback.

**Parameters**

<i>sourceId</i>	The source to use for playback.
<i>numBuffers</i>	The number of buffers to queue.
<i>bufferIds</i>	The IDs of the buffers to queue.

**Returns**

TRUE if the operation is successful.

**4.13.2.100** + (bool) sourceRewind: dummy(ALuint) *sourceId*

Rewind a source.

**Parameters**

<i>sourceId</i>	The ID of the source to rewind.
-----------------	---------------------------------

**Returns**

TRUE if the operation is successful.

**4.13.2.101** + (bool) sourceRewindv: dummy(ALuint\*) *sourceIds* numSources:(ALsizei)  
*numSources*

Rewind a bunch of sources.

**Parameters**

<i>sourceIds</i>	The sources to rewind.
<i>numSources</i>	The number of sources in <i>sourceIds</i> .

**Returns**

TRUE if the operation is successful.

**4.13.2.102** + (bool) sourceStop: dummy(ALuint) *sourceId*

Stop a source.

**Parameters**

<i>sourceId</i>	The ID of the source to stop.
-----------------	-------------------------------

**Returns**

TRUE if the operation is successful.

**4.13.2.103 + (bool) sourceStopv: dummy(ALuint\*) sourceIds numSources:(ALsizei) numSources**

Stop a bunch of sources.

**Parameters**

<i>sourceIds</i>	The sources to stop.
<i>numSources</i>	The number of sources in sourceIds.

**Returns**

TRUE if the operation is successful.

**4.13.2.104 + (bool) sourceUnqueueBuffers: dummy(ALuint) sourceId numBuffers:(ALsizei) numBuffers bufferIds:(ALuint\*) bufferIds**

Unqueue previously queued buffers.

**Parameters**

<i>sourceId</i>	The source the buffers were previously queued in.
<i>numBuffers</i>	The number of buffers to unqueue.
<i>bufferIds</i>	The IDs of the buffers to unqueue.

**Returns**

TRUE if the operation is successful.

**4.13.2.105 + (bool) speedOfSound: dummy(ALfloat) value**

Set the speed of sound.

**Parameters**

<i>value</i>	The value to set.
--------------	-------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.106 + (bool) startCapture: dummy(ALCdevice\*) device**

Start capturing audio data.

**Parameters**

<i>device</i>	The device to capture on.
---------------	---------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.107 + (bool) stopCapture: dummy(ALCdevice\*) device**

Stop capturing audio data.

**Parameters**

<i>device</i>	The device capturing audio data.
---------------	----------------------------------

**Returns**

TRUE if the operation was successful.

**4.13.2.108 + (void) suspendContext: dummy(ALCcontext\*) context**

Suspend a context.

**Parameters**

<i>context</i>	The context to suspend.
----------------	-------------------------

**Returns**

TRUE if the operation was successful.

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

- ALWrapper.h
- ALWrapper.m

## 4.14 IOSVersion Class Reference

Reports the version of iOS being run on the current device.

```
#import <IOSVersion.h>
```

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)

*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

## Properties

- float [version](#)

*The version of iOS being run on the current device as a float in the format x.yy.*

### 4.14.1 Detailed Description

Reports the version of iOS being run on the current device.

### 4.14.2 Member Function Documentation

#### 4.14.2.1 - IOSVersion: dummy(IOSVersion)

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (IOSVersion\*) **sharedInstance**: Get the shared singleton instance.
- (void) **purgeSharedInstance**: Purge (deallocate) the shared instance.

### 4.14.3 Property Documentation

#### 4.14.3.1 - (float) version [read, assign]

The version of iOS being run on the current device as a float in the format x.yy.

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

- IOSVersion.h

## 4.15 NSMutableArray Class Reference

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

- NSMutableArray+WeakReferences.m

## 4.16 OAL\_AsyncALBufferLoadOperation Class Reference

(INTERNAL USE) NSOperation for loading audio files asynchronously.

### Public Member Functions

- (id) - [initWithUrl:reduceToMono:target:selector:](#)  
(INTERNAL USE) Initialize an Asynchronous Operation.

### Static Public Member Functions

- (id) + [operationWithURL:reduceToMono:target:selector:](#)  
(INTERNAL USE) Create a new Asynchronous Operation.

### Protected Attributes

- NSURL \* [url](#)  
The URL of the sound file to play.
- bool [reduceToMono](#)  
If true, reduce the sample to mono.
- id [target](#)  
The target to inform when the operation completes.
- SEL [selector](#)  
The selector to call when the operation completes.

#### 4.16.1 Detailed Description

(INTERNAL USE) NSOperation for loading audio files asynchronously.

#### 4.16.2 Member Function Documentation

4.16.2.1 - (id) initWithUrl: dummy(NSURL\*) url reduceToMono:(bool) *reduceToMono* target:(id) *target* selector:(SEL) *selector*

(INTERNAL USE) Initialize an Asynchronous Operation.

#### Parameters

<i>url</i>	the URL containing the sound file.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).
<i>target</i>	the target to inform when the operation completes.
<i>selector</i>	the selector to call when the operation completes.

**4.16.2.2** + (id) *operationWithURL: dummy(NSURL\*) url reduceToMono:(bool) reduceToMono target:(id) target selector:(SEL) selector*

(INTERNAL USE) Create a new Asynchronous Operation.

#### Parameters

<i>url</i>	the URL containing the sound file.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).
<i>target</i>	the target to inform when the operation completes.
<i>selector</i>	the selector to call when the operation completes.

### 4.16.3 Member Data Documentation

**4.16.3.1** - (bool) *reduceToMono* [protected]

If true, reduce the sample to mono.

**4.16.3.2** - (SEL) *selector* [protected]

The selector to call when the operation completes.

**4.16.3.3** - (id) *target* [protected]

The target to inform when the operation completes.

**4.16.3.4** - (NSURL\*) *url* [protected]

The URL of the sound file to play.

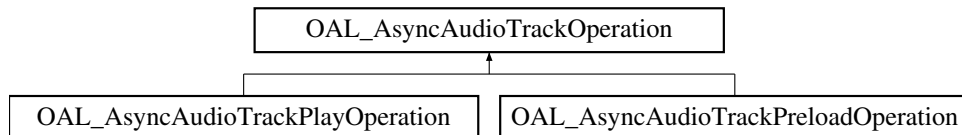
The documentation for this class was generated from the following file:

- OpenALManager.m

## 4.17 OAL\_AsyncAudioTrackOperation Class Reference

(INTERNAL USE) NSOperation for running an audio operation asynchronously.

Inheritance diagram for OAL\_AsyncAudioTrackOperation:



### Public Member Functions

- (id) - [initWithTrack:url:seekTime:target:selector:](#)  
(INTERNAL USE) Initialize an Asynchronous Operation.

### Static Public Member Functions

- (id) + [operationWithTrack:url:seekTime:target:selector:](#)  
(INTERNAL USE) Create a new Asynchronous Operation.

### Protected Attributes

- OALAudioTrack \* [audioTrack](#)  
The audio track object to perform the operation on.
- NSURL \* [url](#)  
The URL of the sound file to play.
- NSTimeInterval [seekTime](#)  
The seekTime of the sound file.
- id [target](#)  
The target to inform when the operation completes.
- SEL [selector](#)  
The selector to call when the operation completes.

#### 4.17.1 Detailed Description

(INTERNAL USE) NSOperation for running an audio operation asynchronously.



## 4.17.2 Member Function Documentation

**4.17.2.1** - (id) initWithTrack: dummy(OALAudioTrack\*) track url:(NSURL\*) url seekTime:(NSTimeInterval) seekTime target:(id) target selector:(SEL) selector

(INTERNAL USE) Initialize an Asynchronous Operation.

### Parameters

<i>track</i>	the audio track to perform the operation on.
<i>seekTime</i>	the position in the file to start playing at.
<i>url</i>	the URL containing the sound file.
<i>target</i>	the target to inform when the operation completes.
<i>selector</i>	the selector to call when the operation completes.

**4.17.2.2** + (id) operationWithTrack: dummy(OALAudioTrack\*) track url:(NSURL\*) url seekTime:(NSTimeInterval) seekTime target:(id) target selector:(SEL) selector

(INTERNAL USE) Create a new Asynchronous Operation.

### Parameters

<i>track</i>	the audio track to perform the operation on.
<i>seekTime</i>	the position in the file to start playing at.
<i>url</i>	the URL containing the sound file.
<i>target</i>	the target to inform when the operation completes.
<i>selector</i>	the selector to call when the operation completes.

## 4.17.3 Member Data Documentation

**4.17.3.1** - (OALAudioTrack\*) audioTrack [protected]

The audio track object to perform the operation on.

**4.17.3.2** - (NSTimeInterval) seekTime [protected]

The seekTime of the sound file.

**4.17.3.3** - (SEL) selector [protected]

The selector to call when the operation completes.

**4.17.3.4** - (id) target [protected]

The target to inform when the operation completes.

#### 4.17.3.5 - (NSURL\*) url [protected]

The URL of the sound file to play.

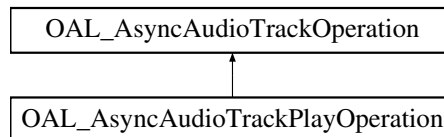
The documentation for this class was generated from the following file:

- OALAudioTrack.m

### 4.18 OAL\_AsyncAudioTrackPlayOperation Class Reference

(INTERNAL USE) NSOperation for playing an audio file asynchronously.

Inheritance diagram for OAL\_AsyncAudioTrackPlayOperation:



#### Public Member Functions

- (id) - initWithTrack:url:loops:target:selector:  
(INTERNAL USE) Initialize an asynchronous play operation.

#### Static Public Member Functions

- (id) + operationWithTrack:url:loops:target:selector:  
(INTERNAL USE) Create an asynchronous play operation.

#### Protected Attributes

- NSInteger loops  
The number of times to loop during playback.

#### 4.18.1 Detailed Description

(INTERNAL USE) NSOperation for playing an audio file asynchronously.

## 4.18.2 Member Function Documentation

**4.18.2.1** - (id) initWithTrack: dummy(OALAudioTrack\*) track url:(NSURL\*) url loops:(NSInteger) loops target:(id) target selector:(SEL) selector

(INTERNAL USE) Initialize an asynchronous play operation.

### Parameters

<i>track</i>	the audio track to perform the operation on.
<i>url</i>	The URL of the file to play.
<i>loops</i>	The number of times to loop playback (-1 = forever).
<i>target</i>	The target to inform when playback finishes.
<i>selector</i>	the selector to call when playback finishes.

### Returns

The initialized operation.

**4.18.2.2** + (id) operationWithTrack: dummy(OALAudioTrack\*) track url:(NSURL\*) url loops:(NSInteger) loops target:(id) target selector:(SEL) selector

(INTERNAL USE) Create an asynchronous play operation.

### Parameters

<i>track</i>	the audio track to perform the operation on.
<i>url</i>	The URL of the file to play.
<i>loops</i>	The number of times to loop playback (-1 = forever).
<i>target</i>	The target to inform when playback finishes.
<i>selector</i>	the selector to call when playback finishes.

### Returns

a new operation.

## 4.18.3 Member Data Documentation

**4.18.3.1** - (NSInteger) loops [protected]

The number of times to loop during playback.

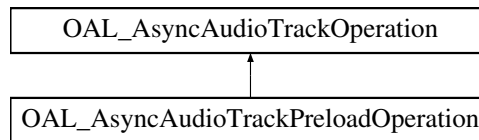
The documentation for this class was generated from the following file:

- OALAudioTrack.m

## 4.19 OAL\_AsyncAudioTrackPreloadOperation Class Reference

(INTERNAL USE) NSOperation for preloading an audio file asynchronously.

Inheritance diagram for OAL\_AsyncAudioTrackPreloadOperation:



### 4.19.1 Detailed Description

(INTERNAL USE) NSOperation for preloading an audio file asynchronously.

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

- OALAudioTrack.m

## 4.20 <OAL\_GainProtocol> Protocol Reference

(INTERNAL USE) Protocol to keep the compiler happy.

### Properties

- float [gain](#)

*The gain (volume), represented as a float from 0.0 to 1.0.*

### 4.20.1 Detailed Description

(INTERNAL USE) Protocol to keep the compiler happy.

### 4.20.2 Property Documentation

#### 4.20.2.1 -(float) gain [read, write, assign]

The gain (volume), represented as a float from 0.0 to 1.0.

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

- OALAudioActions.m

## 4.21 <OAL\_PanProtocol> Protocol Reference

(INTERNAL USE) Protocol to keep the compiler happy.

### Properties

- float [pan](#)

*The pan, represented as a float from -1.0 to 1.0.*

### 4.21.1 Detailed Description

(INTERNAL USE) Protocol to keep the compiler happy.

### 4.21.2 Property Documentation

#### 4.21.2.1 - (float) pan [read, write, assign]

The pan, represented as a float from -1.0 to 1.0.

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

- OALAudioActions.m

## 4.22 <OAL\_PitchProtocol> Protocol Reference

(INTERNAL USE) Protocol to keep the compiler happy.

### Properties

- float [pitch](#)

*The pitch, represented as a float with 1.0 representing normal pitch.*

### 4.22.1 Detailed Description

(INTERNAL USE) Protocol to keep the compiler happy.

### 4.22.2 Property Documentation

#### 4.22.2.1 - (float) pitch [read, write, assign]

The pitch, represented as a float with 1.0 representing normal pitch.

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

- OALAudioActions.m

## 4.23 <OAL\_PositionProtocol> Protocol Reference

(INTERNAL USE) Protocol to keep the compiler happy.

### Properties

- [ALPoint position](#)

*The position in 3D space.*

### 4.23.1 Detailed Description

(INTERNAL USE) Protocol to keep the compiler happy.

### 4.23.2 Property Documentation

#### 4.23.2.1 - (ALPoint) position [read, write, assign]

The position in 3D space.

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

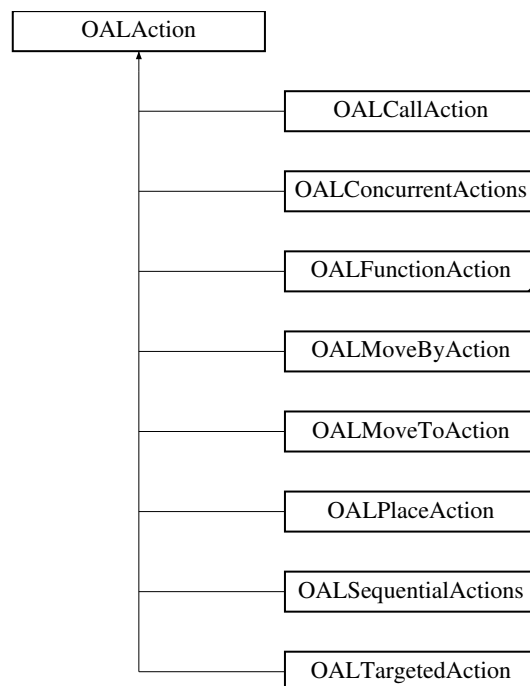
- OALAudioActions.m

## 4.24 OALAction Class Reference

Represents an action that can be performed on an object.

```
#import <OALAction.h>
```

Inheritance diagram for OALAction:



### Public Member Functions

- (id) - [initWithDuration:](#)  
*Initialize an action.*
- (void) - [runWithTarget:](#)  
*Run this action on a target.*
- (void) - [prepareWithTarget:](#)  
*Called by runWithTarget to do any final preparations before running.*
- (void) - [startAction](#)  
*Called by runWithTarget to start the action running.*
- (void) - [updateCompletion:](#)  
*Called by [OALActionManager](#) to update this action's progress.*
- (void) - [stopAction](#)  
*Stop this action.*

### Protected Attributes

- bool [runningInManager](#)

If *TRUE*, this action is running via [OALActionManager](#).

## Properties

- id [target](#)

*The target to perform the action on.*

- float [duration](#)

*The duration of the action, in seconds.*

- float [elapsed](#)

*The amount of time that has elapsed for this action, in seconds.*

- bool [running](#)

*If true, the action is currently running.*

### 4.24.1 Detailed Description

Represents an action that can be performed on an object.

### 4.24.2 Member Function Documentation

#### 4.24.2.1 - (id) initWithDuration: dummy(float) duration

Initialize an action.

##### Parameters

<i>duration</i>	The duration of this action in seconds.
-----------------	---

##### Returns

The initialized action.

#### 4.24.2.2 - (void) prepareWithTarget: dummy(id) target

Called by runWithTraget to do any final preparations before running.

Subclasses must ensure that duration is valid when this method returns.

##### Parameters

<i>target</i>	The target to run the action on.
---------------	----------------------------------



#### 4.24.2.3 - (void) runWithTarget: dummy(id) target

Run this action on a target.

##### Parameters

<i>target</i>	The target to run the action on.
---------------	----------------------------------

#### 4.24.2.4 - (void) startAction

Called by runWithTarget to start the action running.

#### 4.24.2.5 - (void) stopAction

Stop this action.

#### 4.24.2.6 - (void) updateCompletion: dummy(float) proportionComplete

Called by [OALActionManager](#) to update this action's progress.

##### Parameters

<i>proportion-Complete</i>	The proportion of this action's duration that has elapsed.
----------------------------	--

### 4.24.3 Member Data Documentation

#### 4.24.3.1 - (bool) runningInManager [protected]

If TRUE, this action is running via [OALActionManager](#).

### 4.24.4 Property Documentation

#### 4.24.4.1 - (float) duration [read, assign]

The duration of the action, in seconds.

#### 4.24.4.2 - (float) elapsed [read, write, assign]

The amount of time that has elapsed for this action, in seconds.

#### 4.24.4.3 - (bool) running [read, assign]

If true, the action is currently running.

#### 4.24.4.4 - (id) target [read, assign]

The target to perform the action on.

WEAK REFERENCE.

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

- OALAction.h
- OALAction.m

## 4.25 OALActionManager Class Reference

Manages all ObjectAL actions.

```
#import <OALActionManager.h>
```

### Public Member Functions

- (void) - [stopAllActions](#)  
*Stops ALL running actions on ALL targets.*
- (void) - [notifyActionStarted:](#)  
*(INTERNAL USE) Used by [OALAction](#) to announce that it is starting.*
- (void) - [notifyActionStopped:](#)  
*(INTERNAL USE) Used by [OALAction](#) to announce that it is stopping.*
- (void) - [doResetTimeDelta:](#)  
*Resets the time delta in cases where proper time delta calculations become impossible.*

### Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)  
*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

### Protected Attributes

- [NSMutableArray](#) \* [targets](#)  
*All targets that have actions running on them (id).*
- [NSMutableArray](#) \* [targetActions](#)

*Parallel array to "targets", maintaining a list of all actions per target (NSMutableArray\*)*

- NSMutableArray \* [actionsToAdd](#)

*All actions that are to be added on the next pass (OALAction\*)*

- NSMutableArray \* [actionsToRemove](#)

*All actions that are to be removed on the next pass (OALAction\*)*

- NSTimer \* [stepTimer](#)

*The timer which we use to update the actions.*

- uint64\_t [lastTimestamp](#)

*The last time that was recorded.*

### 4.25.1 Detailed Description

Manages all ObjectAL actions.

### 4.25.2 Member Function Documentation

#### 4.25.2.1 - (void) doResetTimeDelta: dummy(NSNotification \*) *notification*

Resets the time delta in cases where proper time delta calculations become impossible.

#### 4.25.2.2 - (void) notifyActionStarted: dummy(OALAction\*) *action*

(INTERNAL USE) Used by [OALAction](#) to announce that it is starting.

##### Parameters

<i>action</i>	The action that is starting.
---------------	------------------------------

#### 4.25.2.3 - (void) notifyActionStopped: dummy(OALAction\*) *action*

(INTERNAL USE) Used by [OALAction](#) to announce that it is stopping.

##### Parameters

<i>action</i>	The action that is stopping.
---------------	------------------------------

#### 4.25.2.4 - (void) stopAllActions

Stops ALL running actions on ALL targets.

#### 4.25.2.5 - OALActionManager: dummy(OALActionManager)

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALAudioSupport\*) sharedInstance: Get the shared singleton instance.

- (void) purgeSharedInstance: Purge (deallocate) the shared instance.

### 4.25.3 Member Data Documentation

#### 4.25.3.1 - (NSMutableArray\*) actionsToAdd [protected]

All actions that are to be added on the next pass (OALAction\*)

#### 4.25.3.2 - (NSMutableArray\*) actionsToRemove [protected]

All actions that are to be removed on the next pass (OALAction\*)

#### 4.25.3.3 - (uint64\_t) lastTimestamp [protected]

The last time that was recorded.

#### 4.25.3.4 - (NSTimer\*) stepTimer [protected]

The timer which we use to update the actions.

#### 4.25.3.5 - (NSMutableArray\*) targetActions [protected]

Parallel array to "targets", maintaining a list of all actions per target (NSMutableArray\*)

#### 4.25.3.6 - (NSMutableArray\*) targets [protected]

All targets that have actions running on them (id).

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

- OALActionManager.h
- OALActionManager.m

## 4.26 OALAudioFile Class Reference

Maintains an open audio file and allows loading data from that file into new [ALBuffer](#) objects.

```
#import <OALAudioFile.h>
```

## Public Member Functions

- (id) - [initWithUrl:reduceToMono:](#)  
*Initialize this object with the audio file at the specified URL.*
- (void \*) - [audioDataWithStartFrame:numFrames:bufferSize:](#)  
*Read audio data from this file into a new buffer.*
- (ALBuffer \*) - [bufferNamed:startFrame:numFrames:](#)  
*Create a new [ALBuffer](#) with the contents of this file.*
- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*

## Static Public Member Functions

- (OALAudioFile \*) + [fileWithUrl:reduceToMono:](#)  
*Open the audio file at the specified URL.*
- (ALBuffer \*) + [bufferFromUrl:reduceToMono:](#)  
*Convenience method to load the entire contents of a URL into a new [ALBuffer](#).*

## Protected Attributes

- AudioStreamBasicDescription [streamDescription](#)  
*A description of the audio data in this file.*
- ExtAudioFileRef [fileHandle](#)  
*The OS specific file handle.*
- UInt32 [originalChannelsPerFrame](#)  
*The actual number of channels in the audio data if not reducing to mono.*

## Properties

- NSURL \* [url](#)  
*The URL of the audio file.*
- AudioStreamBasicDescription \* [streamDescription](#)  
*A description of the audio data in this file.*
- SInt64 [totalFrames](#)  
*The total number of audio frames in this file.*
- bool [reduceToMono](#)  
*If YES, reduce any stereo data to mono (stereo samples don't support panning or positional audio).*

### 4.26.1 Detailed Description

Maintains an open audio file and allows loading data from that file into new [ALBuffer](#) objects.

### 4.26.2 Member Function Documentation

**4.26.2.1 - (void \*) audioDataWithStartFrame: dummy(SInt64) startFrame numFrames:(SInt64) numFrames bufferSize:(UInt32\*) bufferSize**

Read audio data from this file into a new buffer.

#### Parameters

<i>startFrame</i>	The starting audio frame to read data from.
<i>numFrames</i>	The number of frames to read.
<i>bufferSize</i>	On successful return, contains the size of the returned buffer, in bytes.

#### Returns

The audio data or nil on error. You are responsible for calling free() on the data.

**4.26.2.2 + (ALBuffer \*) bufferFromUrl: dummy(NSURL\*) url reduceToMono:(bool) reduceToMono**

Convenience method to load the entire contents of a URL into a new [ALBuffer](#).

#### Parameters

<i>url</i>	The URL to open the audio file from.
<i>reduceToMono</i>	If YES, reduce any stereo track to mono (stereo samples don't support panning or positional audio).

**Returns**

an [ALBuffer](#) object.

**4.26.2.3** - (ALBuffer \*) **bufferNamed:** *dummy*(NSString\*) *name* **startFrame:**(SInt64) *startFrame* **numFrames:**(SInt64) *numFrames*

Create a new [ALBuffer](#) with the contents of this file.

**Parameters**

<i>name</i>	The name to be given to this <a href="#">ALBuffer</a> .
<i>startFrame</i>	The starting audio frame to read data from.
<i>numFrames</i>	The number of frames to read.

**Returns**

a new [ALBuffer](#) containing the audio data.

**4.26.2.4** - (void) **close**

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

**4.26.2.5** - (void) **closeOSResources**

(INTERNAL USE) Close any resources belonging to the OS.

**4.26.2.6** + (OALAudioFile \*) **fileWithURL:** *dummy*(NSURL\*) *url* **reduceToMono:**(bool) *reduceToMono*

Open the audio file at the specified URL.

**Parameters**

<i>url</i>	The URL to open the audio file from.
<i>reduceToMono</i>	If YES, reduce any stereo track to mono (stereo samples don't support panning or positional audio).

**Returns**

a new audio file object.

**4.26.2.7** - (id) **initWithUrl:** *dummy*(NSURL\*) *url* **reduceToMono:**(bool) *reduceToMono*

Initialize this object with the audio file at the specified URL.

**Parameters**

<i>url</i>	The URL to open the audio file from.
<i>reduce-ToMono</i>	If YES, reduce any stereo track to mono (stereo samples don't support panning or positional audio).

**Returns**

the initialized audio file object.

**4.26.3 Member Data Documentation****4.26.3.1 - (ExtAudioFileRef) fileHandle [protected]**

The OS specific file handle.

**4.26.3.2 - (UInt32) originalChannelsPerFrame [protected]**

The actual number of channels in the audio data if not reducing to mono.

**4.26.3.3 - (AudioStreamBasicDescription \*) streamDescription [protected]**

A description of the audio data in this file.

**4.26.4 Property Documentation****4.26.4.1 - (bool) reduceToMono [read, write, assign]**

If YES, reduce any stereo data to mono (stereo samples don't support panning or positional audio).

**4.26.4.2 - (AudioStreamBasicDescription\*) streamDescription [read, assign]**

A description of the audio data in this file.

**4.26.4.3 - (SInt64) totalFrames [read, assign]**

The total number of audio frames in this file.

**4.26.4.4 - (NSURL \*) url [read, assign]**

The URL of the audio file.

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

- OALAudioFile.h



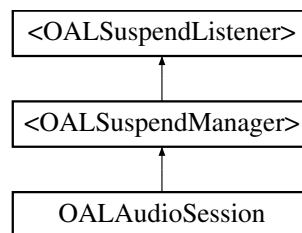
- OALAudioFile.m

## 4.27 OALAudioSession Class Reference

Handles the audio session and interrupts.

```
#import <OALAudioSession.h>
```

Inheritance diagram for OALAudioSession:



### Public Member Functions

- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (void) - [forceEndInterruption](#)  
*Force an interrupt end.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*
- (UInt32) - [getIntProperty:](#)  
*(INTERNAL USE) Get an AudioSession property.*
- (Float32) - [getFloatProperty:](#)  
*(INTERNAL USE) Get an AudioSession property.*
- (NSString \*) - [getStringProperty:](#)  
*(INTERNAL USE) Get an AudioSession property.*
- (void) - [setIntProperty:value:](#)  
*(INTERNAL USE) Set an AudioSession property.*
- (void) - [setAudioMode](#)  
*(INTERNAL USE) Set the Audio Session category and properties based on current settings.*

- (void) - [updateFromAudioSessionCategory](#)  
*(INTERNAL USE) Update settings to be compatible with the current audio session category.*
- (void) - [updateFromFlags](#)  
*(INTERNAL USE) Update the audio session category to be compatible with the current settings.*
- (void) - [setSuspended:](#)  
*(INTERNAL USE) Called by SuspendHandler.*
- (void) - [onAudioError:](#)  
*(INTERNAL USE) Called when an audio error is signalled via notification.*

### Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)  
*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

### Protected Attributes

- bool [audioSessionWasActive](#)  
*If true, the audio session was active when the interrupt occurred.*
- [OALSuspendHandler](#) \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*
- NSDate \* [lastResetTime](#)  
*Marks the last time the audio session was reset due to error.*

### Properties

- NSString \* [audioSessionCategory](#)  
*The current audio session category.*
- bool [allowIpod](#)  
*If YES, allow ipod music to continue playing (NOT SUPPORTED ON THE SIMULATOR).*
- bool [ipodDucking](#)  
*If YES, ipod music will duck (lower in volume) when the audio session activates.*

- bool [useHardwareIfAvailable](#)  
*Determines what to do if no other application is playing audio and allowIpod = YES (NOT SUPPORTED ON THE SIMULATOR).*
- bool [honorSilentSwitch](#)  
*If true, mute when backgrounded, screen locked, or the ringer switch is turned off (NOT SUPPORTED ON THE SIMULATOR).*
- bool [handleInterruptions](#)  
*If true, automatically handle interruptions.*
- id< AVAudioSessionDelegate > [audioSessionDelegate](#)  
*Delegate that will receive all audio session events.*
- bool [ipodPlaying](#)  
*If true, another application (usually iPod) is playing music.*
- bool [audioSessionActive](#)  
*If true, the audio session is active.*
- float [hardwareVolume](#)  
*Get the device's final hardware output volume, as controlled by the volume button on the side of the device.*
- bool [hardwareMuted](#)  
*Check if the hardware mute switch is on (not supported on the simulator).*
- NSString \* [audioRoute](#)  
*Check what hardware route the audio is taking, such as "Speaker" or "Headphone" (not supported on the simulator).*

### 4.27.1 Detailed Description

Handles the audio session and interrupts.

### 4.27.2 Member Function Documentation

#### 4.27.2.1 - (void) close

Close any OS resources in use by this object.

This will close the audio session.

#### 4.27.2.2 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

#### 4.27.2.3 - (void) forceEndInterruption

Force an interrupt end.

This can be useful in cases where a buggy OS fails to end an interrupt.

Be VERY CAREFUL when using this!

#### 4.27.2.4 - (Float32) getFloatProperty: dummy(AudioSessionPropertyID) *property*

(INTERNAL USE) Get an AudioSession property.

##### Parameters

<i>property</i>	The property to get.
-----------------	----------------------

##### Returns

The property's value.

#### 4.27.2.5 - (UInt32) getIntProperty: dummy(AudioSessionPropertyID) *property*

(INTERNAL USE) Get an AudioSession property.

##### Parameters

<i>property</i>	The property to get.
-----------------	----------------------

##### Returns

The property's value.

#### 4.27.2.6 - (NSString\*) getStringProperty: dummy(AudioSessionPropertyID) *property*

(INTERNAL USE) Get an AudioSession property.

##### Parameters

<i>property</i>	The property to get.
-----------------	----------------------

##### Returns

The property's value.

**4.27.2.7 - (void) onAudioError: dummy(NSNotification \*) notification**

(INTERNAL USE) Called when an audio error is signalled via notification.

**4.27.2.8 - (void) setAudioMode**

(INTERNAL USE) Set the Audio Session category and properties based on current settings.

**4.27.2.9 - (void) setIntProperty: dummy(AudioSessionPropertyID) property value:(UInt32) value**

(INTERNAL USE) Set an AudioSession property.

**Parameters**

<i>property</i>	The property to set.
<i>value</i>	The value to set this property to.

**4.27.2.10 - (void) setSuspended: dummy(bool) value**

(INTERNAL USE) Called by SuspendHandler.

**4.27.2.11 - OALAudioSession: dummy(OALAudioSession)**

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALAudioSupport\*) sharedInstance: Get the shared singleton instance.

- (void) purgeSharedInstance: Purge (deallocate) the shared instance.

**4.27.2.12 - (void) updateFromAudioSessionCategory**

(INTERNAL USE) Update settings to be compatible with the current audio session category.

**4.27.2.13 - (void) updateFromFlags**

(INTERNAL USE) Update the audio session category to be compatible with the current settings.

### 4.27.3 Member Data Documentation

#### 4.27.3.1 - (bool) `audioSessionWasActive` [protected]

If true, the audio session was active when the interrupt occurred.

#### 4.27.3.2 - (NSDate\*) `lastResetTime` [protected]

Marks the last time the audio session was reset due to error.

This is used to avoid getting stuck in a rapid-fire reset-error loop.

#### 4.27.3.3 - (OALSuspendHandler\*) `suspendHandler` [protected]

Handles suspending and interrupting for this object.

### 4.27.4 Property Documentation

#### 4.27.4.1 - (bool) `allowIpod` [read, write, assign]

If YES, allow ipod music to continue playing (NOT SUPPORTED ON THE SIMULATOR).

Note: If this is enabled, and another app is playing music, background audio playback will use the SOFTWARE codecs, NOT hardware.

If `allowIpod` = NO, the application will ALWAYS use hardware decoding.

See also

[useHardwareIfAvailable](#)

Default value: YES

#### 4.27.4.2 - (NSString \*) `audioRoute` [read, assign]

Check what hardware route the audio is taking, such as "Speaker" or "Headphone" (not supported on the simulator).

#### 4.27.4.3 - (bool) `audioSessionActive` [read, write, assign]

If true, the audio session is active.

#### 4.27.4.4 - (NSString \*) `audioSessionCategory` [read, write, retain]

The current audio session category.

If this value is explicitly set, the other session properties "allowIpod", "useHardwareIfAvailable", "honorSilentSwitch", and "ipodDucking" may be modified to remain compatible with the category.

**See also**

AVAudioSessionCategory

Default value: nil

**4.27.4.5 - (id< AVAudioSessionDelegate >) audioSessionDelegate** [read, write, assign]

Delegate that will receive all audio session events.

**4.27.4.6 - (bool) handleInterruptions** [read, write, assign]

If true, automatically handle interruptions.

Default value: YES

**4.27.4.7 - (bool) hardwareMuted** [read, assign]

Check if the hardware mute switch is on (not supported on the simulator).

Note: If headphones are plugged in, hardwareMuted will always return FALSE regardless of the switch state.

**4.27.4.8 - (float) hardwareVolume** [read, assign]

Get the device's final hardware output volume, as controlled by the volume button on the side of the device.

**4.27.4.9 - (bool) honorSilentSwitch** [read, write, assign]

If true, mute when backgrounded, screen locked, or the ringer switch is turned off (NOT SUPPORTED ON THE SIMULATOR).

Default value: YES

**4.27.4.10 - (bool) ipodDucking** [read, write, assign]

If YES, ipod music will duck (lower in volume) when the audio session activates.

Default value: NO

#### 4.27.4.11 - (bool) ipodPlaying [read, assign]

If true, another application (usually iPod) is playing music.

#### 4.27.4.12 - (bool) useHardwareIfAvailable [read, write, assign]

Determines what to do if no other application is playing audio and allowIpod = YES (NOT SUPPORTED ON THE SIMULATOR).

If NO, the application will ALWAYS use software decoding. The advantage to this is that the user can background your application and then start audio playing from another application. If useHardwareIfAvailable = YES, the user won't be able to do this.

If this is set to YES, the application will use hardware decoding if no other application is currently playing audio. However, no other application will be able to start playing audio if it wasn't playing already.

Note: This switch has no effect if allowIpod = NO.

#### See also

[allowIpod](#)

Default value: YES

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

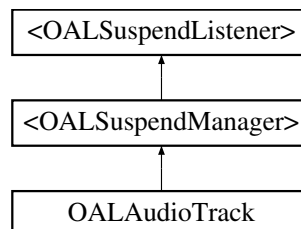
- OALAudioSession.h
- OALAudioSession.m

## 4.28 OALAudioTrack Class Reference

Plays an audio track via AVAudioPlayer.

```
#import <OALAudioTrack.h>
```

Inheritance diagram for OALAudioTrack:



#### Public Member Functions

- (void) - [close](#)



*Close any OS resources in use by this object.*

- (bool) - [preloadUrl:](#)  
*Preload the contents of a URL for playback.*
- (bool) - [preloadUrl:seekTime:](#)  
*Preload the contents of a URL for playback.*
- (bool) - [preloadFile:](#)  
*Preload the contents of a file for playback.*
- (bool) - [preloadFile:seekTime:](#)  
*Preload the contents of a file for playback.*
- (bool) - [preloadUrlAsync:target:selector:](#)  
*Asynchronously preload the contents of a URL for playback.*
- (bool) - [preloadUrlAsync:seekTime:target:selector:](#)  
*Asynchronously preload the contents of a URL for playback.*
- (bool) - [preloadFileAsync:target:selector:](#)  
*Asynchronously preload the contents of a file for playback.*
- (bool) - [preloadFileAsync:seekTime:target:selector:](#)  
*Asynchronously preload the contents of a file for playback.*
- (bool) - [playUrl:](#)  
*Play the contents of a URL once.*
- (bool) - [playUrl:loops:](#)  
*Play the contents of a URL and loop the specified number of times.*
- (bool) - [playFile:](#)  
*Play the contents of a file once.*
- (bool) - [playFile:loops:](#)  
*Play the contents of a file and loop the specified number of times.*
- (void) - [playUrlAsync:target:selector:](#)  
*Play the contents of a URL asynchronously once.*
- (void) - [playUrlAsync:loops:target:selector:](#)  
*Play the contents of a URL asynchronously and loop the specified number of times.*
- (void) - [playFileAsync:target:selector:](#)  
*Play the contents of a file asynchronously once.*

- (void) - [playFileAsync:loops:target:selector:](#)  
*Play the contents of a file asynchronously and loop the specified number of times.*
- (bool) - [play](#)  
*Play the currently loaded audio track.*
- (bool) - [playAtTime:](#)  
*Plays a sound asynchronously, starting at a specified point in the audio output device's timeline.*
- (void) - [stop](#)  
*Stop playing and stop all operations.*
- (void) - [fadeTo:duration:target:selector:](#)  
*Fade to the specified gain value.*
- (void) - [stopFade](#)  
*Stop the currently running fade operation, if any.*
- (void) - [panTo:duration:target:selector:](#)  
*Pan to the specified pan value.*
- (void) - [stopPan](#)  
*Stop the currently running pan operation, if any.*
- (void) - [stopActions](#)  
*Stop any internal fade or pan actions.*
- (void) - [clear](#)  
*Unload and clear all audio data, stop playing, and stop all operations.*
- (void) - [updateMeters](#)  
*Updates the metering system to give current values.*
- (float) - [averagePowerForChannel:](#)  
*Gives the average power for a given channel, in decibels, for the sound being played.*
- (float) - [peakPowerForChannel:](#)  
*Gives the peak power for a given channel, in decibels, for the sound being played.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*
- (void) - [setSuspended:](#)  
*(INTERNAL USE) Called by SuspendHandler.*

## Static Public Member Functions

- (id) + [track](#)  
*Create a new audio track.*

## Protected Attributes

- bool [interrupted](#)  
*If YES, this object is interrupted.*
- AVAudioPlayer \* [simulatorPlayerRef](#)  
*When the simulator is running (and the playback fix is in use), player will be copied to here, and then player set to nil.*
- NSOperationQueue \* [operationQueue](#)  
*Operation queue for running asynchronous operations.*
- OALAction \* [gainAction](#)  
*The current action being applied to gain.*
- OALAction \* [panAction](#)  
*The current action being applied to pan.*
- OALSuspendHandler \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*

## Properties

- NSURL \* [currentlyLoadedUrl](#)  
*The URL of the currently loaded audio data.*
- id< AVAudioPlayerDelegate > [delegate](#)  
*Optional object that will receive notifications for decoding errors, audio interruptions (such as an incoming phone call), and playback completion.*
- float [gain](#)  
*The gain (volume) for playback (0.0 - 1.0, where 1.0 = no attenuation).*
- float [volume](#)  
*The volume (alias to gain) for playback (0.0 - 1.0, where 1.0 = no attenuation).*
- float [pan](#)  
*Pan value (-1.0 = far left, 1.0 = far right).*

- bool [muted](#)  
*If true, audio track is muted.*
- bool [autoPreload](#)  
*If true, automatically preload again when playback stops.*
- bool [preloaded](#)  
*If true, audio track is in preloaded state.*
- NSInteger [numberOfLoops](#)  
*The number of times to loop playback (-1 = forever).*
- bool [paused](#)  
*If true, pause playback.*
- AVAudioPlayer \* [player](#)  
*Access to the underlying AVAudioPlayer object.*
- bool [playing](#)  
*If true, the audio player is currently playing.*
- NSTimeInterval [currentTime](#)  
*The current playback position in seconds from the start of the sound.*
- NSTimeInterval [deviceCurrentTime](#)  
*The value of this property increases monotonically while an audio player is playing or paused.*
- NSTimeInterval [duration](#)  
*The duration, in seconds, of the currently loaded sound.*
- NSInteger [numberOfChannels](#)  
*The number of channels in the currently loaded sound.*
- bool [meteringEnabled](#)  
*If true, metering is enabled.*

#### 4.28.1 Detailed Description

Plays an audio track via AVAudioPlayer. Unlike AVAudioPlayer, however, it can be re-used to play another file. Interruptions can be handled by OALAudioSupport (enabled by default).

## 4.28.2 Member Function Documentation

### 4.28.2.1 - (float) averagePowerForChannel: dummy(NSUInteger) channelNumber

Gives the average power for a given channel, in decibels, for the sound being played.

0 dB indicates maximum power (full scale).

-160 dB indicates minimum power (near silence).

If the signal provided to the audio player exceeds full scale, then the value may be  $> 0$ .

**Note:** The value returned is in reference to when `updateMeters` was last called. You must call `updateMeters` again before calling this method to get a current value.

#### Parameters

<i>channel-Number</i>	The channel to get the value from. For mono or left, use 0. For right, use 1.
-----------------------	---

#### Returns

the average power for the channel.

### 4.28.2.2 - (void) clear

Unload and clear all audio data, stop playing, and stop all operations.

### 4.28.2.3 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

### 4.28.2.4 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

### 4.28.2.5 - (void) fadeTo: dummy(float) gain duration:(float) duration target:(id) target selector:(SEL) selector

Fade to the specified gain value.

#### Parameters

<i>gain</i>	The gain to fade to.
<i>duration</i>	The duration of the fade operation in seconds.
<i>target</i>	The target to notify when the fade completes (can be nil).
<i>selector</i>	The selector to call when the fade completes. The selector must accept a single parameter, which will be the object that performed the fade.

#### 4.28.2.6 - (void) panTo: dummy(float) *pan* duration:(float) *duration* target:(id) *target* selector:(SEL) *selector*

Pan to the specified pan value.

**Note:** This will have no effect on iOS versions prior to 4.0.

##### Parameters

<i>pan</i>	The value to pan to.
<i>duration</i>	The duration of the pan operation in seconds.
<i>target</i>	The target to notify when the pan completes (can be nil).
<i>selector</i>	The selector to call when the pan completes. The selector must accept a single parameter, which will be the object that performed the pan.

#### 4.28.2.7 - (float) peakPowerForChannel: dummy(NSUInteger) *channelNumber*

Gives the peak power for a given channel, in decibels, for the sound being played.

0 dB indicates maximum power (full scale).

-160 dB indicates minimum power (near silence).

If the signal provided to the audio player exceeds full scale, then the value may be  $> 0$ .

**Note:** The value returned is in reference to when `updateMeters` was last called. You must call `updateMeters` again before calling this method to get a current value.

##### Parameters

<i>channel-Number</i>	The channel to get the value from. For mono or left, use 0. For right, use 1.
-----------------------	---

##### Returns

the average power for the channel.

#### 4.28.2.8 - (bool) play

Play the currently loaded audio track.

##### Returns

TRUE if the operation was successful.

#### 4.28.2.9 - (bool) playAtTime: dummy(NSTimeInterval) *time*

Plays a sound asynchronously, starting at a specified point in the audio output device's timeline.

**Note:** This will have no effect on iOS versions prior to 4.0.

**4.28.2.10 - (bool) playFile: dummy(NSString\*) path**

Play the contents of a file once.

**Parameters**

<i>path</i>	The file containing the sound data.
-------------	-------------------------------------

**Returns**

TRUE if the operation was successful.

**4.28.2.11 - (bool) playFile: dummy(NSString\*) path loops:(NSInteger) loops**

Play the contents of a file and loop the specified number of times.

**Parameters**

<i>path</i>	The file containing the sound data.
<i>loops</i>	The number of times to loop playback (-1 = forever)

**Returns**

TRUE if the operation was successful.

**4.28.2.12 - (void) playFileAsync: dummy(NSString\*) path loops:(NSInteger) loops target:(id) target selector:(SEL) selector**

Play the contents of a file asynchronously and loop the specified number of times.

**Parameters**

<i>path</i>	The file containing the sound data.
<i>loops</i>	The number of times to loop playback (-1 = forever)
<i>target</i>	the target to inform when playing has started.
<i>selector</i>	the selector to call when playing has started.

**4.28.2.13 - (void) playFileAsync: dummy(NSString\*) path target:(id) target selector:(SEL) selector**

Play the contents of a file asynchronously once.

**Parameters**

<i>path</i>	The file containing the sound data.
<i>target</i>	the target to inform when playing has started.
<i>selector</i>	the selector to call when playing has started.

**4.28.2.14 - (bool) playUrl: dummy(NSURL\*) url**

Play the contents of a URL once.

**Parameters**

<i>url</i>	The URL containing the sound data.
------------	------------------------------------

**Returns**

TRUE if the operation was successful.

**4.28.2.15 - (bool) playUrl: dummy(NSURL\*) url loops:(NSInteger) loops**

Play the contents of a URL and loop the specified number of times.

**Parameters**

<i>url</i>	The URL containing the sound data.
<i>loops</i>	The number of times to loop playback (-1 = forever)

**Returns**

TRUE if the operation was successful.

**4.28.2.16 - (void) playUrlAsync: dummy(NSURL\*) url loops:(NSInteger) loops target:(id) target selector:(SEL) selector**

Play the contents of a URL asynchronously and loop the specified number of times.

**Parameters**

<i>url</i>	The URL containing the sound data.
<i>loops</i>	The number of times to loop playback (-1 = forever)
<i>target</i>	the target to inform when playing has started.
<i>selector</i>	the selector to call when playing has started.

**4.28.2.17 - (void) playUrlAsync: dummy(NSURL\*) url target:(id) target selector:(SEL) selector**

Play the contents of a URL asynchronously once.

**Parameters**

<i>url</i>	The URL containing the sound data.
<i>target</i>	the target to inform when playing has started.
<i>selector</i>	the selector to call when playing has started.



**4.28.2.18 - (bool) preloadFile: dummy(NSString\*) path**

Preload the contents of a file for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>path</i>	The file containing the sound data.
-------------	-------------------------------------

**Returns**

TRUE if the operation was successful.

**4.28.2.19 - (bool) preloadFile: dummy(NSString\*) path seekTime:(NSTimeInterval) seekTime**

Preload the contents of a file for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>path</i>	The file containing the sound data.
<i>seekTime</i>	The position in the file to start playing at.

**Returns**

TRUE if the operation was successful.

**4.28.2.20 - (bool) preloadFileAsync: dummy(NSString\*) path seekTime:(NSTimeInterval) seekTime target:(id) target selector:(SEL) selector**

Asynchronously preload the contents of a file for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>path</i>	The file containing the sound data.
<i>seekTime</i>	The position in the file to start playing at.
<i>target</i>	the target to inform when preparation is complete.
<i>selector</i>	the selector to call when preparation is complete.

**Returns**

TRUE if the operation was successfully queued.

**4.28.2.21 - (bool) preloadFileAsync: dummy(NSString\*) path target:(id) target selector:(SEL) selector**

Asynchronously preload the contents of a file for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>path</i>	The file containing the sound data.
<i>target</i>	the target to inform when preparation is complete.
<i>selector</i>	the selector to call when preparation is complete.

**Returns**

TRUE if the operation was successfully queued.

**4.28.2.22 - (bool) preloadUrl: dummy(NSURL\*) url**

Preload the contents of a URL for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>url</i>	The URL containing the sound data.
------------	------------------------------------

**Returns**

TRUE if the operation was successful.

**4.28.2.23 - (bool) preloadUrl: dummy(NSURL\*) url seekTime:(NSTimeInterval) seekTime**

Preload the contents of a URL for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>url</i>	The URL containing the sound data.
<i>seekTime</i>	The position in the file to start playing at.

**Returns**

TRUE if the operation was successful.

**4.28.2.24 - (bool) preloadUrlAsync: dummy(NSURL\*) url seekTime:(NSTimeInterval) seekTime target:(id) target selector:(SEL) selector**

Asynchronously preload the contents of a URL for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>url</i>	The URL containing the sound data.
<i>seekTime</i>	The position in the file to start playing at.
<i>target</i>	the target to inform when preparation is complete.
<i>selector</i>	the selector to call when preparation is complete.

**Returns**

TRUE if the operation was successfully queued.

**4.28.2.25 - (bool) preloadUrlAsync: dummy(NSURL\*) url target:(id) target selector:(SEL) selector**

Asynchronously preload the contents of a URL for playback.

Once the audio data is preloaded, you can call "play" to play it.

**Parameters**

<i>url</i>	The URL containing the sound data.
<i>target</i>	the target to inform when preparation is complete.
<i>selector</i>	the selector to call when preparation is complete.

**Returns**

TRUE if the operation was successfully queued.

**4.28.2.26 - (void) setSuspended: dummy(bool) value**

(INTERNAL USE) Called by SuspendHandler.

**4.28.2.27 - (void) stop**

Stop playing and stop all operations.

**4.28.2.28 - (void) stopActions**

Stop any internal fade or pan actions.

**4.28.2.29 - (void) stopFade**

Stop the currently running fade operation, if any.

#### 4.28.2.30 - (void) stopPan

Stop the currently running pan operation, if any.

**Note:** This will have no effect on iOS versions prior to 4.0.

#### 4.28.2.31 + (id) track

Create a new audio track.

##### Returns

A new audio track.

#### 4.28.2.32 - (void) updateMeters

Updates the metering system to give current values.

You must call this method before calling `averagePowerForChannel` or `peakPowerForChannel` in order to get current values.

### 4.28.3 Member Data Documentation

#### 4.28.3.1 - (OALAction\*) gainAction [protected]

The current action being applied to gain.

#### 4.28.3.2 - (bool) interrupted [protected]

If YES, this object is interrupted.

**Note:** This property must NOT be set by the user!

Reimplemented from [<OALSuspendListener>](#).

#### 4.28.3.3 - (NSOperationQueue\*) operationQueue [protected]

Operation queue for running asynchronous operations.

**Note:** Only one asynchronous operation is allowed at a time.

#### 4.28.3.4 - (OALAction\*) panAction [protected]

The current action being applied to pan.

**4.28.3.5 - (AVAudioPlayer\*) simulatorPlayerRef** [protected]

When the simulator is running (and the playback fix is in use), player will be copied to here, and then player set to nil.

This prevents other code from inadvertently raising the volume and starting playback.

**4.28.3.6 - (OALSuspendHandler\*) suspendHandler** [protected]

Handles suspending and interrupting for this object.

**4.28.4 Property Documentation****4.28.4.1 - (bool) autoPreload** [read, write, assign]

If true, automatically preload again when playback stops.

**4.28.4.2 - (NSURL \*) currentlyLoadedUrl** [read, assign]

The URL of the currently loaded audio data.

**4.28.4.3 - (NSTimeInterval) currentTime** [read, write, assign]

The current playback position in seconds from the start of the sound.

You can set this to change the playback position, whether it is currently playing or not.

**4.28.4.4 - (id< AVAudioPlayerDelegate >) delegate** [read, write, assign]

Optional object that will receive notifications for decoding errors, audio interruptions (such as an incoming phone call), and playback completion.

**Note:** [OALAudioTrack](#) keeps a WEAK reference to delegate, so make sure you clear it when your object is going to be deallocated.

**4.28.4.5 - (NSTimeInterval) deviceCurrentTime** [read, assign]

The value of this property increases monotonically while an audio player is playing or paused.

If more than one audio player is connected to the audio output device, device time continues incrementing as long as at least one of the players is playing or paused.

If the audio output device has no connected audio players that are either playing or paused, device time reverts to 0.

Use this property to indicate “now” when calling the `playAtTime:` instance method. By configuring multiple audio players to play at a specified offset from `deviceCurrentTime`, you can perform precise synchronization—as described in the discussion for that method.

**Note:** This will have no effect on iOS versions prior to 4.0.

#### 4.28.4.6 - (NSTimeInterval) duration [read, assign]

The duration, in seconds, of the currently loaded sound.

#### 4.28.4.7 - (float) gain [read, write, assign]

The gain (volume) for playback (0.0 - 1.0, where 1.0 = no attenuation).

#### 4.28.4.8 - (bool) meteringEnabled [read, write, assign]

If true, metering is enabled.

#### 4.28.4.9 - (bool) muted [read, write, assign]

If true, audio track is muted.

#### 4.28.4.10 - (NSInteger) numberOfChannels [read, assign]

The number of channels in the currently loaded sound.

#### 4.28.4.11 - (NSInteger) numberOfLoops [read, write, assign]

The number of times to loop playback (-1 = forever).

**Note:** This value will be ignored, and get changed when you call the various `playXX` methods. Only “play” will use the current value of “numberOfLoops”.

#### 4.28.4.12 - (float) pan [read, write, assign]

Pan value (-1.0 = far left, 1.0 = far right).

**Note:** This will have no effect on iOS versions prior to 4.0.

#### 4.28.4.13 - (bool) paused [read, write, assign]

If true, pause playback.

**4.28.4.14 - (AVAudioPlayer \*) player** [read, assign]

Access to the underlying AVAudioPlayer object.

WARNING: Be VERY careful when accessing this, as some methods could cause it to fall out of sync with [OALAudioTrack](#) (particularly play/pause/stop methods).

**4.28.4.15 - (bool) playing** [read, assign]

If true, the audio player is currently playing.

If true, background music is currently playing.

We need to maintain our own value because AVAudioPlayer will sometimes say it's not playing when it actually is.

**4.28.4.16 - (bool) preloaded** [read, assign]

If true, audio track is in preloaded state.

**4.28.4.17 - (float) volume** [read, write, assign]

The volume (alias to gain) for playback (0.0 - 1.0, where 1.0 = no attenuation).

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

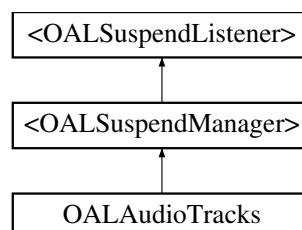
- OALAudioTrack.h
- OALAudioTrack.m

## 4.29 OALAudioTracks Class Reference

Keeps track of all AudioTrack objects.

```
#import <OALAudioTracks.h>
```

Inheritance diagram for OALAudioTracks:



## Public Member Functions

- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (void) - [notifyTrackInitializing:](#)  
*(INTERNAL USE) Notify that a track is initializing.*
- (void) - [notifyTrackDeallocating:](#)  
*(INTERNAL USE) Notify that a track is deallocating.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)  
*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

## Protected Attributes

- [NSMutableArray](#) \* [tracks](#)  
*All instantiated audio tracks.*
- [OALSuspendHandler](#) \* [suspendHandler](#)  
*Handles suspending and interrupting for this object.*

## Properties

- bool [paused](#)  
*Pauses/unpauses all audio tracks.*
- bool [muted](#)  
*Mutes/unmutes all audio tracks.*
- [NSArray](#) \* [tracks](#)  
*All instantiated audio tracks.*



### 4.29.1 Detailed Description

Keeps track of all AudioTrack objects.

### 4.29.2 Member Function Documentation

#### 4.29.2.1 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

#### 4.29.2.2 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

#### 4.29.2.3 - (void) notifyTrackDeallocating: dummy(OALAudioTrack\*) track

(INTERNAL USE) Notify that a track is deallocating.

#### 4.29.2.4 - (void) notifyTrackInitializing: dummy(OALAudioTrack\*) track

(INTERNAL USE) Notify that a track is initializing.

#### 4.29.2.5 - OALAudioTracks: dummy(OALAudioTracks)

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALAudioTracks\*) sharedInstance: Get the shared singleton instance.

- (void) purgeSharedInstance: Purge (deallocate) the shared instance.

### 4.29.3 Member Data Documentation

#### 4.29.3.1 - (OALSuspendHandler\*) suspendHandler [protected]

Handles suspending and interrupting for this object.

#### 4.29.3.2 - (NSMutableArray\*) tracks [protected]

All instantiated audio tracks.

## 4.29.4 Property Documentation

### 4.29.4.1 `-(bool) muted` [read, write, assign]

Mutes/unmutes all audio tracks.

### 4.29.4.2 `-(bool) paused` [read, write, assign]

Pauses/unpauses all audio tracks.

### 4.29.4.3 `-(NSArray*) tracks` [read, assign]

All instantiated audio tracks.

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

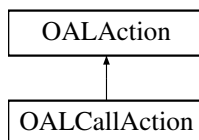
- OALAudioTracks.h
- OALAudioTracks.m

## 4.30 OALCallAction Class Reference

Calls a selector on a target.

```
#import <OALUtilityActions.h>
```

Inheritance diagram for OALCallAction:



### Public Member Functions

- (id) - `initWithCallTarget:selector:`  
*Initialize an action.*
- (id) - `initWithCallTarget:selector:withObject:`  
*Initialize an action.*
- (id) - `initWithCallTarget:selector:withObject:withObject:`  
*Initialize an action.*

## Static Public Member Functions

- (id) + [actionWithCallTarget:selector:](#)  
*Create an action.*
- (id) + [actionWithCallTarget:selector:withObject:](#)  
*Create an action.*
- (id) + [actionWithCallTarget:selector:withObject:withObject:](#)  
*Create an action.*

## Protected Attributes

- id [callTarget](#)  
*The target to call the selector on.*
- SEL [selector](#)  
*The selector to invoke.*
- int [numObjects](#)  
*The number of parameters which will be passed to the selector.*
- id [object1](#)  
*The first object to pass to the selector, if any.*
- id [object2](#)  
*The second object to pass to the selector, if any.*

### 4.30.1 Detailed Description

Calls a selector on a target. This action will ignore whatever target it is run against, and will invoke the selector on the target specified at creation time.

### 4.30.2 Member Function Documentation

#### 4.30.2.1 + (id) actionWithCallTarget: dummy(id) *callTarget* selector:(SEL) *selector*

Create an action.

#### Parameters

<i>callTarget</i>	The target to call.
<i>selector</i>	The selector to invoke.

**Returns**

A new action.

**4.30.2.2 + (id) actionWithCallTarget: dummy(id) callTarget selector:(SEL) selector  
withObject:(id) object**

Create an action.

**Parameters**

<i>callTarget</i>	The target to call.
<i>selector</i>	The selector to invoke.
<i>object</i>	The object to pass to the selector.

**Returns**

A new action.

**4.30.2.3 + (id) actionWithCallTarget: dummy(id) callTarget selector:(SEL) selector  
withObject:(id) firstObject withObject:(id) secondObject**

Create an action.

**Parameters**

<i>callTarget</i>	The target to call.
<i>selector</i>	The selector to invoke.
<i>firstObject</i>	The first object to pass to the selector.
<i>secondObject</i>	The second object to pass to the selector.

**Returns**

A new action.

**4.30.2.4 - (id) initWithCallTarget: dummy(id) callTarget selector:(SEL) selector**

Initialize an action.

**Parameters**

<i>callTarget</i>	The target to call.
<i>selector</i>	The selector to invoke.

**Returns**

The initialized action.

#### 4.30.2.5 - (id) initWithCallTarget: dummy(id) callTarget selector:(SEL) selector withObject:(id) object

Initialize an action.

##### Parameters

<i>callTarget</i>	The target to call.
<i>selector</i>	The selector to invoke.
<i>object</i>	The object to pass to the selector.

##### Returns

Initialize an action.

#### 4.30.2.6 - (id) initWithCallTarget: dummy(id) callTarget selector:(SEL) selector withObject:(id) firstObject withObject:(id) secondObject

Initialize an action.

##### Parameters

<i>callTarget</i>	The target to call.
<i>selector</i>	The selector to invoke.
<i>firstObject</i>	The first object to pass to the selector.
<i>secondObject</i>	The second object to pass to the selector.

##### Returns

The initialized action.

### 4.30.3 Member Data Documentation

#### 4.30.3.1 - (id) callTarget [protected]

The target to call the selector on.

#### 4.30.3.2 - (int) numObjects [protected]

The number of parameters which will be passed to the selector.

#### 4.30.3.3 - (id) object1 [protected]

The first object to pass to the selector, if any.

#### 4.30.3.4 - (id) object2 [protected]

The second object to pass to the selector, if any.

#### 4.30.3.5 - (SEL) selector [protected]

The selector to invoke.

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

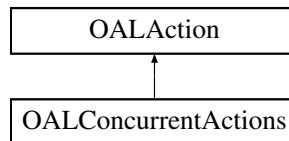
- OALUtilityActions.h
- OALUtilityActions.m

## 4.31 OALConcurrentActions Class Reference

A set of actions that get run concurrently.

```
#import <OALUtilityActions.h>
```

Inheritance diagram for OALConcurrentActions:



### Public Member Functions

- (id) - [initWithActions:](#)  
*Initialize an action.*

### Static Public Member Functions

- (id) + [actions:](#)  
*Create an action.*
- (id) + [actionsFromArray:](#)  
*Create an action.*

## Protected Attributes

- [NSMutableArray](#) \* [pDurations](#)  
*The durations of the actions.*
- [NSMutableArray](#) \* [actionsWithDuration](#)  
*A list of actions that have duration > 0.*

## Properties

- [NSMutableArray](#) \* [actions](#)  
*The actions which will be run.*

### 4.31.1 Detailed Description

A set of actions that get run concurrently.

### 4.31.2 Member Function Documentation

#### 4.31.2.1 + (id) actions: dummy(OALAction\*) actions, NS\_REQUIRES\_NIL\_TERMINATION

Create an action.

##### Parameters

<i>actions</i>	The comma separated list of actions.
<i>NS_REQUIRES_NIL_TERMINATION</i>	List of actions must be terminated by a nil.

##### Returns

A new set of concurrent actions.

#### 4.31.2.2 + (id) actionsFromArray: dummy(NSArray\*) actions

Create an action.

##### Parameters

<i>actions</i>	The actions to run.
----------------	---------------------

**Returns**

A new set of concurrent actions.

**4.31.2.3 - (id) initWithActions: dummy(NSArray\*) actions**

Initialize an action.

**Parameters**

<i>actions</i>	The actions to run.
----------------	---------------------

**Returns**

The initialized set of concurrent actions.

**4.31.3 Member Data Documentation****4.31.3.1 - (NSMutableArray\*) actionsWithDuration [protected]**

A list of actions that have duration > 0.

**4.31.3.2 - (NSMutableArray\*) pDurations [protected]**

The durations of the actions.

**4.31.4 Property Documentation****4.31.4.1 - (NSMutableArray \*) actions [read, write, retain]**

The actions which will be run.

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

- OALUtilityActions.h
- OALUtilityActions.m

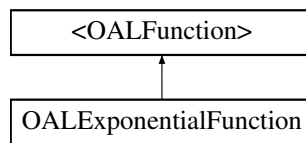
**4.32 OALExponentialFunction Class Reference**

Changes slowly at the start, and quickly at the end.

```
#import <OALFunction.h>
```

Inheritance diagram for OALExponentialFunction:





## Static Public Member Functions

- (id) + [function](#)

*Generate an instance of this function.*

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)

*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

### 4.32.1 Detailed Description

Changes slowly at the start, and quickly at the end.

```

#
#
#
#
#
#
##
###
####
#####
#####
#####

```

### 4.32.2 Member Function Documentation

#### 4.32.2.1 + (id) function

Generate an instance of this function.

#### Returns

An instance of this function.

#### 4.32.2.2 - OALExponentialFunction: dummy(OALExponentialFunction)

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALExponentialFunction\*) sharedInstance: Get the shared singleton instance.

- (void) purgeSharedInstance: Purge (deallocate) the shared instance.

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

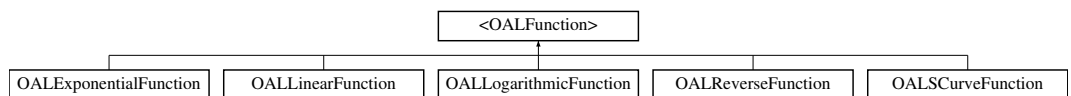
- OALFunction.h
- OALFunction.m

### 4.33 <OALFunction> Protocol Reference

A function takes a value from 0.0 to 1.0 and returns another value from 0.0 to 1.0.

```
#import <OALFunction.h>
```

Inheritance diagram for <OALFunction>:



#### Public Member Functions

- (float) - [valueForInput](#):  
*Calculate the function value.*

#### 4.33.1 Detailed Description

A function takes a value from 0.0 to 1.0 and returns another value from 0.0 to 1.0.

#### 4.33.2 Member Function Documentation

##### 4.33.2.1 - (float) valueForInput: dummy(float) inputValue

Calculate the function value.

#### Parameters

<i>inputValue</i>	A value from 0.0 to 1.0
-------------------	-------------------------

**Returns**

The resulting value, which will also be from 0.0 to 1.0.

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

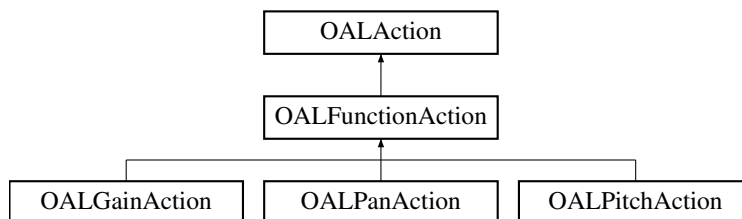
- OALFunction.h

**4.34 OALFunctionAction Class Reference**

An action that applies a function to the proportionComplete parameter in [update] before applying the result to the target.

```
#import <OALAction.h>
```

Inheritance diagram for OALFunctionAction:

**Public Member Functions**

- (id) - [initWithDuration:endValue:](#)  
*Initialize an action using the default function.*
- (id) - [initWithDuration:endValue:function:](#)  
*Initialize an action.*
- (id) - [initWithDuration:startValue:endValue:function:](#)  
*Initialize an action.*

**Static Public Member Functions**

- (id) + [actionWithDuration:endValue:](#)  
*Create a new action using the default function.*
- (id) + [actionWithDuration:endValue:function:](#)  
*Create a new action.*
- (id) + [actionWithDuration:startValue:endValue:function:](#)

Create a new action.

- (id< [OALFunction](#), NSObject >) + [defaultFunction](#)

Get the function that this action would use by default if none was specified.

## Protected Attributes

- float [lowValue](#)

The lowest value that will ever be set over the course of this function.

- float [delta](#)

The difference between the lowest and highest value.

- [OALReverseFunction](#) \* [reverseFunction](#)

The reverse function, if any.

- id< [OALFunction](#), NSObject > [realFunction](#)

The basic function that will be applied normally, or reversed.

## Properties

- id< [OALFunction](#), NSObject > [function](#)

The function that will be applied.

- float [startValue](#)

The value that the property in the target will hold at the start of the action.

- float [endValue](#)

The value that the property in the target will hold at the end of the action.

### 4.34.1 Detailed Description

An action that applies a function to the proportionComplete parameter in [update] before applying the result to the target. This allows things like exponential and s-curve functions when applying gain transitions, for example.

### 4.34.2 Member Function Documentation

#### 4.34.2.1 + (id) initWithDuration: dummy(float) duration endValue:(float) endValue

Create a new action using the default function.

The start value will be the current value of the target this action is applied to.

**Parameters**

<i>duration</i>	The duration of this action in seconds.
<i>endValue</i>	The "ending" value that this action will converge upon when setting the target's property.

**Returns**

A new action.

#### 4.34.2.2 + (id) initWithDuration: dummy(float) *duration* endValue:(float) *endValue* function:(id<OALFunction,NSObject>) *function*

Create a new action.

The start value will be the current value of the target this action is applied to.

**Parameters**

<i>duration</i>	The duration of this action in seconds.
<i>endValue</i>	The "ending" value that this action will converge upon when setting the target's property.
<i>function</i>	The function to apply in this action's update method.

**Returns**

A new action.

#### 4.34.2.3 + (id) initWithDuration: dummy(float) *duration* startValue:(float) *startValue* endValue:(float) *endValue* function:(id<OALFunction,NSObject>) *function*

Create a new action.

**Parameters**

<i>duration</i>	The duration of this action in seconds.
<i>startValue</i>	The "starting" value that this action will diverge from when setting the target's property. If NAN, use the current value from the target.
<i>endValue</i>	The "ending" value that this action will converge upon when setting the target's property.
<i>function</i>	The function to apply in this action's update method.

**Returns**

A new action.

#### 4.34.2.4 + (id< OALFunction, NSObject >) defaultFunction

Get the function that this action would use by default if none was specified.

**4.34.2.5 - (id) initWithDuration: dummy(float) duration endValue:(float) endValue**

Initialize an action using the default function.

The start value will be the current value of the target this action is applied to.

**Parameters**

<i>duration</i>	The duration of this action in seconds.
<i>endValue</i>	The "ending" value that this action will converge upon when setting the target's property.

**Returns**

The initialized action.

**4.34.2.6 - (id) initWithDuration: dummy(float) duration endValue:(float) endValue function:(id<OALFunction,NSObject>) function**

Initialize an action.

The start value will be the current value of the target this action is applied to.

**Parameters**

<i>duration</i>	The duration of this action in seconds.
<i>endValue</i>	The "ending" value that this action will converge upon when setting the target's property.
<i>function</i>	The function to apply in this action's update method.

**Returns**

The initialized action.

**4.34.2.7 - (id) initWithDuration: dummy(float) duration startValue:(float) startValue endValue:(float) endValue function:(id<OALFunction,NSObject>) function**

Initialize an action.

**Parameters**

<i>duration</i>	The duration of this action in seconds.
<i>startValue</i>	The "starting" value that this action will diverge from when setting the target's property. If NAN, use the current value from the target.
<i>endValue</i>	The "ending" value that this action will converge upon when setting the target's property.
<i>function</i>	The function to apply in this action's update method.

**Returns**

The initialized action.

### 4.34.3 Member Data Documentation

#### 4.34.3.1 - (float) delta [protected]

The difference between the lowest and highest value.

#### 4.34.3.2 - (float) lowValue [protected]

The lowest value that will ever be set over the course of this function.

#### 4.34.3.3 - (id<OALFunction,NSObject>) realFunction [protected]

The basic function that will be applied normally, or reversed.

#### 4.34.3.4 - (OALReverseFunction\*) reverseFunction [protected]

The reverse function, if any.

When this is not null, the reverse function is used.

### 4.34.4 Property Documentation

#### 4.34.4.1 - (float) endValue [read, write, assign]

The value that the property in the target will hold at the end of the action.

#### 4.34.4.2 - (id< OALFunction, NSObject >) function [read, write, retain]

The function that will be applied.

#### 4.34.4.3 - (float) startValue [read, write, assign]

The value that the property in the target will hold at the start of the action.

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

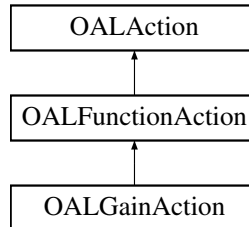
- OALAction.h
- OALAction.m

## 4.35 OALGainAction Class Reference

A function-based action that modifies the target's gain.

```
#import <OALAudioActions.h>
```

Inheritance diagram for OALGainAction:



#### 4.35.1 Detailed Description

A function-based action that modifies the target's gain. The target's gain property is assumed to be a float, accepting values from 0.0 (no sound) to 1.0 (max gain).

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

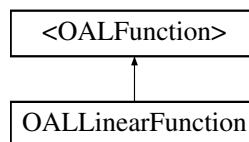
- OALAudioActions.h

### 4.36 OALLinearFunction Class Reference

Function that changes at a constant rate.

```
#import <OALFunction.h>
```

Inheritance diagram for OALLinearFunction:



#### Static Public Member Functions

- (id) + [function](#)  
*Generate an instance of this function.*

#### Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)  
*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*



Function that changes at a constant rate.

A 10x10 grid of hash symbols (#) forming a diagonal line from the bottom-left to the top-right.

#### 4.36.2.1 + (id) function

## Returns

An instance of this function.

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (**OALLinearFunction\***) **sharedInstance**: Get the shared singleton instance.

- **(void) purgeSharedInstance:** Purge (deallocate) the shared instance.

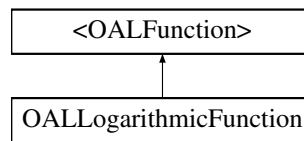
The documentation for this class was generated from the following files:

- OALFunction.h
- OALFunction.m

Changes quickly at the start, and slowly at the end.

```
#import <OALFunction.h>
```

Inheritance diagram for OALLogarithmicFunction:



## Static Public Member Functions

- (id) + [function](#)  
*Generate an instance of this function.*

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)  
*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

### 4.37.1 Detailed Description

Changes quickly at the start, and slowly at the end.

```

#####
#####
####
###
###
#
#
#
#
#
#
#

```

### 4.37.2 Member Function Documentation

#### 4.37.2.1 + (id) function

Generate an instance of this function.

#### Returns

An instance of this function.

#### 4.37.2.2 - OALLogarithmicFunction: dummy(OALLogarithmicFunction)

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALLogarithmicFunction\*) sharedInstance: Get the shared singleton instance.
- (void) purgeSharedInstance: Purge (deallocate) the shared instance.

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

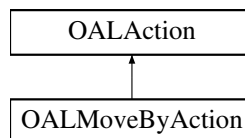
- OALFunction.h
- OALFunction.m

## 4.38 OALMoveByAction Class Reference

Moves the target from its current position by the specified delta over time in 3D space.

```
#import <OALAudioActions.h>
```

Inheritance diagram for OALMoveByAction:



### Public Member Functions

- (id) - initWithDuration:delta:  
*Initialize an action.*
- (id) - initWithUnitsPerSecond:delta:  
*Initialize an action.*

### Static Public Member Functions

- (id) + actionWithDuration:delta:  
*Create a new action.*
- (id) + actionWithUnitsPerSecond:delta:  
*Create a new action.*

## Protected Attributes

- [ALPoint startPoint](#)

*The point this move is starting at.*

## Properties

- [ALPoint delta](#)

*The amount to move the target by.*

- float [unitsPerSecond](#)

*The speed at which to move the target.*

### 4.38.1 Detailed Description

Moves the target from its current position by the specified delta over time in 3D space.

### 4.38.2 Member Function Documentation

#### 4.38.2.1 + (id) actionWithDuration: dummy(float) duration delta:(ALPoint) delta

Create a new action.

##### Parameters

<i>duration</i>	The duration of the move.
<i>delta</i>	The amount to move by.

##### Returns

A new action.

#### 4.38.2.2 + (id) actionWithUnitsPerSecond: dummy(float) unitsPerSecond delta:(ALPoint) delta

Create a new action.

##### Parameters

<i>unitsPerSecond</i>	The rate of movement.
<i>delta</i>	The amount to move by.

##### Returns

A new action.

**4.38.2.3 - (id) initWithDuration: dummy(float) *duration* delta:(ALPoint) *delta***

Initialize an action.

**Parameters**

<i>duration</i>	The duration of the move.
<i>delta</i>	The amount to move by.

**Returns**

The initialized action.

**4.38.2.4 - (id) initWithUnitsPerSecond: dummy(float) *unitsPerSecond* delta:(ALPoint) *delta***

Initialize an action.

**Parameters**

<i>unitsPerSecond</i>	The rate of movement.
<i>delta</i>	The amount to move by.

**Returns**

The initialized action.

**4.38.3 Member Data Documentation****4.38.3.1 - (ALPoint) startPoint [protected]**

The point this move is starting at.

**4.38.4 Property Documentation****4.38.4.1 - (ALPoint) delta [read, write, assign]**

The amount to move the target by.

**4.38.4.2 - (float) unitsPerSecond [read, write, assign]**

The speed at which to move the target.

If this is 0, the target will be moved at the speed determined by duration.

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

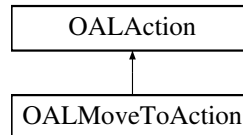
- OALAudioActions.h
- OALAudioActions.m

## 4.39 OALMoveToAction Class Reference

Moves the target from its current position to the specified position over time in 3D space.

```
#import <OALAudioActions.h>
```

Inheritance diagram for OALMoveToAction:



### Public Member Functions

- (id) - [initWithDuration:position:](#)  
*Initialize an action.*
- (id) - [initWithUnitsPerSecond:position:](#)  
*Initialize an action.*

### Static Public Member Functions

- (id) + [actionWithDuration:position:](#)  
*Create a new action.*
- (id) + [actionWithUnitsPerSecond:position:](#)  
*Create a new action.*

### Protected Attributes

- [ALPoint startPoint](#)  
*The point this move is starting at.*
- [ALPoint delta](#)  
*The distance being moved.*

### Properties

- [ALPoint position](#)  
*The position to move the target to.*

- float [unitsPerSecond](#)

*The speed at which to move the target.*

### 4.39.1 Detailed Description

Moves the target from its current position to the specified position over time in 3D space.

### 4.39.2 Member Function Documentation

#### 4.39.2.1 + (id) actionWithDuration: dummy(float) *duration* position:(ALPoint) *position*

Create a new action.

##### Parameters

<i>duration</i>	The duration of the move.
<i>position</i>	The position to move to.

##### Returns

A new action.

#### 4.39.2.2 + (id) actionWithUnitsPerSecond: dummy(float) *unitsPerSecond* position:(ALPoint) *position*

Create a new action.

##### Parameters

<i>unitsPerSecond</i>	The rate of movement.
<i>position</i>	The position to move to.

##### Returns

A new action.

#### 4.39.2.3 - (id) initWithDuration: dummy(float) *duration* position:(ALPoint) *position*

Initialize an action.

##### Parameters

<i>duration</i>	The duration of the move.
<i>position</i>	The position to move to.

**Returns**

The initialized action.

**4.39.2.4 - (id) initWithUnitsPerSecond: dummy(float) unitsPerSecond position:(ALPoint) position**

Initialize an action.

**Parameters**

<i>unitsPerSecond</i>	The rate of movement.
<i>position</i>	The position to move to.

**Returns**

The initialized action.

**4.39.3 Member Data Documentation****4.39.3.1 - (ALPoint) delta [protected]**

The distance being moved.

**4.39.3.2 - (ALPoint) startPoint [protected]**

The point this move is starting at.

**4.39.4 Property Documentation****4.39.4.1 - (ALPoint) position [read, write, assign]**

The position to move the target to.

**4.39.4.2 - (float) unitsPerSecond [read, write, assign]**

The speed at which to move the target.

If this is 0, the target will be moved at the speed determined by duration.

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

- OALAudioActions.h
- OALAudioActions.m

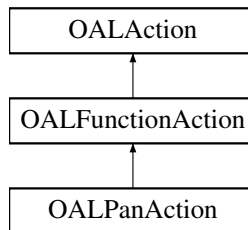


## 4.40 OALPanAction Class Reference

A function-based action that modifies the target's pan.

```
#import <OALAudioActions.h>
```

Inheritance diagram for OALPanAction:



### 4.40.1 Detailed Description

A function-based action that modifies the target's pan. The target's pan property is assumed to be a float, accepting values from -1.0 (max left) to 1.0 (max right).

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

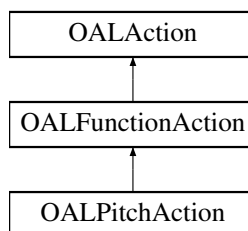
- OALAudioActions.h

## 4.41 OALPitchAction Class Reference

A function-based action that modifies the target's pitch.

```
#import <OALAudioActions.h>
```

Inheritance diagram for OALPitchAction:



### 4.41.1 Detailed Description

A function-based action that modifies the target's pitch. The target's pitch property is assumed to be a float, with 1.0 representing normal pitch, and lower values giving lower pitch.

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

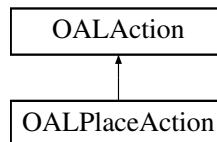
- OALAudioActions.h

## 4.42 OALPlaceAction Class Reference

Places the target at the specified position.

```
#import <OALAudioActions.h>
```

Inheritance diagram for OALPlaceAction:



### Public Member Functions

- (id) - [initWithPosition:](#)  
*Initialize an action with the specified position.*

### Static Public Member Functions

- (id) + [actionWithPosition:](#)  
*Create an action with the specified position.*

### Properties

- [ALPoint position](#)  
*The position where the target will be placed.*

#### 4.42.1 Detailed Description

Places the target at the specified position.

### 4.42.2 Member Function Documentation

#### 4.42.2.1 + (id) actionWithPosition: dummy(ALPoint) *position*

Create an action with the specified position.

##### Parameters

<i>position</i>	The position to place the target at.
-----------------	--------------------------------------

##### Returns

A new action.

#### 4.42.2.2 - (id) initWithPosition: dummy(ALPoint) *position*

Initialize an action with the specified position.

##### Parameters

<i>position</i>	The position to place the target at.
-----------------	--------------------------------------

##### Returns

The initialized action.

### 4.42.3 Property Documentation

#### 4.42.3.1 - (ALPoint) *position* [read, write, assign]

The position where the target will be placed.

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

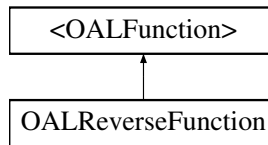
- OALAudioActions.h
- OALAudioActions.m

## 4.43 OALReverseFunction Class Reference

Returns the reverse of another function.

```
#import <OALFunction.h>
```

Inheritance diagram for OALReverseFunction:



## Public Member Functions

- (id) - [initWithFunction:](#)  
*Initialize a reverse function.*

## Static Public Member Functions

- (id) + [functionWithFunction:](#)  
*Create a new reverse function.*

## Properties

- id< [OALFunction](#), NSObject > [function](#)  
*The function which will have its value reversed.*

### 4.43.1 Detailed Description

Returns the reverse of another function. For example, a linear up ramp will become a linear down ramp:

Before:	After:	
##	##	
##	##	
##	##	
##	##	
##	##	
##	##	
##	##	

### 4.43.2 Member Function Documentation

#### 4.43.2.1 + (id) [functionWithFunction:](#) dummy(id<[OALFunction](#), NSObject>) [function](#)

Create a new reverse function.

#### Parameters

<i>function</i>	The function to reverse.
-----------------	--------------------------

**Returns**

the new reversed function.

**4.43.2.2 - (id) initWithFunction: dummy(id<OALFunction, NSObject>) *function***

Initialize a reverse function.

**Parameters**

<i>function</i>	The function to reverse.
-----------------	--------------------------

**Returns**

the initialized reversed function.

**4.43.3 Property Documentation****4.43.3.1 - (id< OALFunction, NSObject >) *function* [read, write, retain]**

The function which will have its value reversed.

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

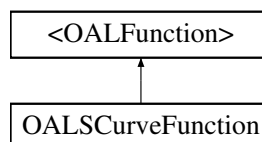
- OALFunction.h
- OALFunction.m

**4.44 OALSCurveFunction Class Reference**

Changes slowly at the start, quickly at the midpoint, then slowly again at the end.

```
#import <OALFunction.h>
```

Inheritance diagram for OALSCurveFunction:

**Static Public Member Functions**

- (id) + [function](#)

*Generate an instance of this function.*

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)

*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

### 4.44.1 Detailed Description

Changes slowly at the start, quickly at the midpoint, then slowly again at the end.

```

#####
####
###
##
#
#
#
#
#
##
###
####

```

### 4.44.2 Member Function Documentation

#### 4.44.2.1 + (id) function

Generate an instance of this function.

##### Returns

An instance of this function.

#### 4.44.2.2 - OALSCurveFunction: dummy(OALSCurveFunction)

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALSCurveFunction\*) **sharedInstance**: Get the shared singleton instance.

- (void) **purgeSharedInstance**: Purge (deallocate) the shared instance.

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

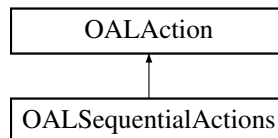
- OALFunction.h
- OALFunction.m

## 4.45 OALSequentialActions Class Reference

A set of actions that get run in sequence.

```
#import <OALUtilityActions.h>
```

Inheritance diagram for OALSequentialActions:



### Public Member Functions

- (id) - [initWithActions:](#)  
*Initialize an action.*

### Static Public Member Functions

- (id) + [actions:](#)  
*Create an action.*
- (id) + [actionsFromArray:](#)  
*Create an action.*

### Protected Attributes

- [NSMutableArray](#) \* [pDurations](#)  
*The durations of the actions.*
- uint [actionIndex](#)  
*The index of the action currently being processed.*
- float [pLastComplete](#)  
*The last completeness proportion value acted upon.*
- [OALAction](#) \* [currentAction](#)  
*The current action being processed.*
- float [pCurrentActionDuration](#)  
*The proportional duration of the current action.*

- float [pCurrentActionComplete](#)  
*The proportional completeness of the current action.*

## Properties

- [NSMutableArray](#) \* [actions](#)  
*The actions which will be run.*

### 4.45.1 Detailed Description

A set of actions that get run in sequence.

### 4.45.2 Member Function Documentation

#### 4.45.2.1 + (id) actions: dummy(OALAction\*) actions, NS\_REQUIRES\_NIL\_TERMINATION

Create an action.

##### Parameters

<i>actions</i>	The comma separated list of actions.
<i>NS_REQUIRES_NIL_TERMINATION</i>	List of actions must be terminated by a nil.

##### Returns

A new set of sequential actions.

#### 4.45.2.2 + (id) actionsFromArray: dummy(NSArray\*) actions

Create an action.

##### Parameters

<i>actions</i>	The actions to run.
----------------	---------------------

##### Returns

A new set of sequential actions.



**4.45.2.3 - (id) initWithActions: dummy(NSArray\*) actions**

Initialize an action.

**Parameters**

<i>actions</i>	The actions to run.
----------------	---------------------

**Returns**

The initialized set of sequential actions.

**4.45.3 Member Data Documentation****4.45.3.1 - (uint) actionIndex [protected]**

The index of the action currently being processed.

**4.45.3.2 - (OALAction\*) currentAction [protected]**

The current action being processed.

**4.45.3.3 - (float) pCurrentActionComplete [protected]**

The proportional completeness of the current action.

**4.45.3.4 - (float) pCurrentActionDuration [protected]**

The proportional duration of the current action.

**4.45.3.5 - (NSMutableArray\*) pDurations [protected]**

The durations of the actions.

**4.45.3.6 - (float) pLastComplete [protected]**

The last completeness proportion value acted upon.

**4.45.4 Property Documentation****4.45.4.1 - (NSMutableArray \*) actions [read, write, retain]**

The actions which will be run.

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

- [OALUtilityActions.h](#)
- [OALUtilityActions.m](#)

## 4.46 OALSimpleAudio Class Reference

A simpler interface to the ObjectAL sound library.

```
#import <OALSimpleAudio.h>
```

### Public Member Functions

- (id) - [initWithSources:](#)  
*(INTERNAL USE) Initialize with the specified number of reserved sources.*
- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (bool) - [preloadBg:](#)  
*Preload background music.*
- (bool) - [preloadBg:seekTime:](#)  
*Preload background music.*
- (bool) - [playBg](#)  
*Play whatever background music is preloaded.*
- (bool) - [playBgWithLoop:](#)  
*Play whatever background music is preloaded.*
- (bool) - [playBg:](#)  
*Play the background music at the specified path.*
- (bool) - [playBg:loop:](#)  
*Play the background music at the specified path.*
- (bool) - [playBg:volume:pan:loop:](#)  
*Play the background music at the specified path.*
- (void) - [stopBg](#)  
*Stop the background music playback and rewind.*
- ([ALBuffer \\*](#)) - [preloadEffect:](#)  
*Preload and cache a sound effect for later playback.*
- ([ALBuffer \\*](#)) - [preloadEffect:reduceToMono:](#)

*Preload and cache a sound effect for later playback.*

- (void) - [unloadEffect:](#)  
*Unload a preloaded effect.*
- (void) - [unloadAllEffects](#)  
*Unload all preloaded effects.*
- (id< [ALSoundSource](#) >) - [playEffect:](#)  
*Play a sound effect with volume 1.0, pitch 1.0, pan 0.0, loop NO.*
- (id< [ALSoundSource](#) >) - [playEffect:loop:](#)  
*Play a sound effect with volume 1.0, pitch 1.0, pan 0.0.*
- (id< [ALSoundSource](#) >) - [playEffect:volume:pitch:pan:loop:](#)  
*Play a sound effect.*
- (id< [ALSoundSource](#) >) - [playBuffer:volume:pitch:pan:loop:](#)  
*Play a sound effect from a user-supplied buffer.*
- (void) - [stopAllEffects](#)  
*Stop ALL sound effect playback.*
- (void) - [stopEverything](#)  
*Stop all effects and bg music.*
- (void) - [resetToDefault](#)  
*Reset everything in this object to its default state.*
- (void) - [closeOSResources](#)  
*(INTERNAL USE) Close any resources belonging to the OS.*
- ([ALBuffer](#) \*) - [internalPreloadEffect:reduceToMono:](#)  
*(INTERNAL USE) Preload a sound effect and return the preloaded buffer.*

## Static Public Member Functions

- ([OALSimpleAudio](#) \*) + [sharedInstanceWithSources:](#)  
*Start [OALSimpleAudio](#) with the specified number of reserved sources.*

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)

*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

## Protected Attributes

- [ALDevice](#) \* [device](#)

*The device we are using.*

- [ALContext](#) \* [context](#)

*The context we are using.*

- [ALChannelSource](#) \* [channel](#)

*The sound channel used by this object.*

- [NSMutableDictionary](#) \* [preloadCache](#)

*Cache for preloaded sound samples.*

- [uint](#) [pendingLoadCount](#)

*keeping track of how many effects remain to be loaded*

## Properties

- [bool](#) [allowIpod](#)

*If YES, allow ipod music to continue playing (NOT SUPPORTED ON THE SIMULATOR).*

- [bool](#) [useHardwareIfAvailable](#)

*Determines what to do if no other application is playing audio and allowIpod = YES (NOT SUPPORTED ON THE SIMULATOR).*

- [bool](#) [honorSilentSwitch](#)

*If true, mute when backgrounded, screen locked, or the ringer switch is turned off (NOT SUPPORTED ON THE SIMULATOR).*

- [unsigned int](#) [reservedSources](#)

*The number of sources [OALSimpleAudio](#) is using (max 32 on current iOS devices).*

- [NSURL](#) \* [backgroundTrackURL](#)

*Background audio URL.*

- [OALAudioTrack](#) \* [backgroundTrack](#)

*Audio track to play background music.*

- bool [bgPaused](#)  
*Pauses BG music playback.*
- bool [bgMuted](#)  
*Mutes BG music playback.*
- bool [bgPlaying](#)  
*If true, BG music is currently playing.*
- float [bgVolume](#)  
*Background music playback gain/volume (0.0 - 1.0)*
- bool [effectsPaused](#)  
*Pauses effects playback.*
- bool [effectsMuted](#)  
*Mutes effects playback.*
- float [effectsVolume](#)  
*Master effects gain/volume (0.0 - 1.0)*
- bool [paused](#)  
*Pauses everything.*
- bool [muted](#)  
*Mutes all audio.*
- bool [preloadCacheEnabled](#)  
*Enables/disables the preload cache.*
- NSInteger [preloadCacheCount](#)  
*The number of items currently in the preload cache.*
- bool [manuallySuspended](#)  
*Set to YES to manually suspend the sound system.*
- bool [interrupted](#)  
*If YES, the sound system is interrupted.*
- bool [suspended](#)  
*If YES, the sound system is suspended.*

### 4.46.1 Detailed Description

A simpler interface to the ObjectAL sound library. This singleton can be used alone for simpler audio needs, or in conjunction with user-created audio objects for more advanced needs (as is done in many of the demos).

For sound effects, it initializes OpenAL with the default [ALDevice](#), an [ALContext](#), and an [ALChannelSource](#) consisting of all 32 interruptible [ALSource](#) objects (the maximum currently allowed for iOS). If you want to create your own sources as well, change the `reservedSources` property.

For background audio, it creates a single [OALAudioTrack](#), which will not reserve resources unless used. (you can create more [OALAudioTrack](#) objects for your own use if you want).

This singleton also provides access to the more common configuration options available in [OALAudioSupport](#).

All audio playback commands are delegated either to the [ALChannelSource](#) (for sound effects), or to the [OALAudioTrack](#) (for BG music).

### 4.46.2 Member Function Documentation

#### 4.46.2.1 - (void) close

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

#### 4.46.2.2 - (void) closeOSResources

(INTERNAL USE) Close any resources belonging to the OS.

#### 4.46.2.3 - (id) initWithSources: dummy(int) sources

(INTERNAL USE) Initialize with the specified number of reserved sources.

#### Parameters

<i>sources</i>	the number of sources to reserve when initializing.
----------------	---

#### Returns

The shared instance.

#### 4.46.2.4 - (ALBuffer\*) internalPreloadEffect: dummy(NSString \*) filePath reduceToMono:(bool) reduceToMono

(INTERNAL USE) Preload a sound effect and return the preloaded buffer.

**Parameters**

<i>filePath</i>	The path containing the sound data.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).

**Returns**

The preloaded buffer.

**4.46.2.5 - (bool) playBg**

Play whatever background music is preloaded.

**Returns**

TRUE if the operation was successful.

**4.46.2.6 - (bool) playBg: dummy(NSString\*) path**

Play the background music at the specified path.

If the music has not been preloaded, this method will load the music and then play, incurring a slight delay.

**Note:** only **ONE** background music file may be played or preloaded at a time via [OAL-SimpleAudio](#). If you play or preload another file, the one currently playing will stop.

**Parameters**

<i>path</i>	The path containing the background music.
-------------	---

**Returns**

TRUE if the operation was successful.

**4.46.2.7 - (bool) playBg: dummy(NSString\*) path loop:(bool) loop**

Play the background music at the specified path.

If the music has not been preloaded, this method will load the music and then play, incurring a slight delay.

**Note:** only **ONE** background music file may be played or preloaded at a time via [OAL-SimpleAudio](#). If you play or preload another file, the one currently playing will stop.

**Parameters**

<i>path</i>	The path containing the background music.
<i>loop</i>	If true, loop the bg track.

**Returns**

TRUE if the operation was successful.

**4.46.2.8** - (bool) **playBg:** *dummy*(NSString\*) *filePath* volume:(float) *volume* pan:(float) *pan* loop:(bool) *loop*

Play the background music at the specified path.

If the music has not been preloaded, this method will load the music and then play, incurring a slight delay.

**Note:** only **ONE** background music file may be played or preloaded at a time via [OAL-SimpleAudio](#). If you play or preload another file, the one currently playing will stop. To play multiple audio tracks, create an [OALAudioTrack](#).

**Note:** pan will have no effect when running on iOS versions prior to 4.0.

**Parameters**

<i>filePath</i>	The path containing the sound data.
<i>volume</i>	The volume (gain) to play at (0.0 - 1.0).
<i>pan</i>	Left-right panning (-1.0 = far left, 1.0 = far right) (Only on iOS 4.0+).
<i>loop</i>	If TRUE, the sound will loop until you call "stopBg".

**Returns**

TRUE if the operation was successful.

**4.46.2.9** - (bool) **playBgWithLoop:** *dummy*(bool) *loop*

Play whatever background music is preloaded.

**Parameters**

<i>loop</i>	If true, loop the bg track.
-------------	-----------------------------

**Returns**

TRUE if the operation was successful.

**4.46.2.10** - (id< [ALSoundSource](#) >) **playBuffer:** *dummy*(ALBuffer\*) *buffer* volume:(float) *volume* pitch:(float) *pitch* pan:(float) *pan* loop:(bool) *loop*

Play a sound effect from a user-supplied buffer.

**Parameters**

<i>buffer</i>	The buffer containing the sound data.
<i>volume</i>	The volume (gain) to play at (0.0 - 1.0).



<i>pitch</i>	The pitch to play at (1.0 = normal pitch).
<i>pan</i>	Left-right panning (-1.0 = far left, 1.0 = far right).
<i>loop</i>	If TRUE, the sound will loop until you call "stop" on the returned sound source.

**Returns**

The sound source being used for playback, or nil if an error occurred (You'll need to keep this if you want to be able to stop a looped playback).

**4.46.2.11 - (id< ALSoundSource >) playEffect: dummy(NSString\*) filePath**

Play a sound effect with volume 1.0, pitch 1.0, pan 0.0, loop NO.

The sound will be loaded and cached if it wasn't already.

**Parameters**

<i>filePath</i>	The path containing the sound data.
-----------------	-------------------------------------

**Returns**

The sound source being used for playback, or nil if an error occurred.

**4.46.2.12 - (id< ALSoundSource >) playEffect: dummy(NSString\*) filePath loop:(bool) loop**

Play a sound effect with volume 1.0, pitch 1.0, pan 0.0.

The sound will be loaded and cached if it wasn't already.

**Parameters**

<i>filePath</i>	The path containing the sound data.
<i>loop</i>	If TRUE, the sound will loop until you call "stop" on the returned sound source.

**Returns**

The sound source being used for playback, or nil if an error occurred.

**4.46.2.13 - (id< ALSoundSource >) playEffect: dummy(NSString\*) filePath volume:(float) volume pitch:(float) pitch pan:(float) pan loop:(bool) loop**

Play a sound effect.

The sound will be loaded and cached if it wasn't already.

**Parameters**

<i>filePath</i>	The path containing the sound data.
<i>volume</i>	The volume (gain) to play at (0.0 - 1.0).
<i>pitch</i>	The pitch to play at (1.0 = normal pitch).
<i>pan</i>	Left-right panning (-1.0 = far left, 1.0 = far right).
<i>loop</i>	If TRUE, the sound will loop until you call "stop" on the returned sound source.

**Returns**

The sound source being used for playback, or nil if an error occurred (You'll need to keep this if you want to be able to stop a looped playback).

**4.46.2.14 - (bool) preloadBg: dummy(NSString\*) path**

Preload background music.

**Note:** only **ONE** background music file may be played or preloaded at a time via [OAL-SimpleAudio](#). If you play or preload another file, the one currently playing will stop.

**Parameters**

<i>path</i>	The path containing the background music.
-------------	---

**Returns**

TRUE if the operation was successful.

**4.46.2.15 - (bool) preloadBg: dummy(NSString\*) path seekTime:(NSTimeInterval) seekTime**

Preload background music.

**Note:** only **ONE** background music file may be played or preloaded at a time via [OAL-SimpleAudio](#). If you play or preload another file, the one currently playing will stop.

**Parameters**

<i>path</i>	The path containing the background music.
<i>seekTime</i>	the position in the file to start playing at.

**Returns**

TRUE if the operation was successful.

**4.46.2.16 - (ALBuffer \*) preloadEffect: dummy(NSString\*) filePath**

Preload and cache a sound effect for later playback.

**Parameters**

<i>filePath</i>	The path containing the sound data.
-----------------	-------------------------------------

**4.46.2.17 - (ALBuffer \*) preloadEffect: dummy(NSString\*) filePath reduceToMono:(bool) reduceToMono**

Preload and cache a sound effect for later playback.

**Parameters**

<i>filePath</i>	The path containing the sound data.
<i>reduceToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).

**4.46.2.18 - (void) resetToDefault**

Reset everything in this object to its default state.

**4.46.2.19 + (OALSimpleAudio \*) sharedInstanceWithSources: dummy(int) sources**

Start [OALSimpleAudio](#) with the specified number of reserved sources.

Call this initializer if you want to use [OALSimpleAudio](#), but keep some of the device's audio sources (there are 32 in total) for your own use.

**Note:** This method must be called ONLY ONCE, *BEFORE* any attempt is made to access the shared instance. To change the reserved sources after instantiation, modify reservedSources.

**Parameters**

<i>sources</i>	the number of sources <a href="#">OALSimpleAudio</a> will reserve for itself.
----------------	---

**Returns**

The shared instance.

**4.46.2.20 - (void) stopAllEffects**

Stop ALL sound effect playback.

**4.46.2.21 - (void) stopBg**

Stop the background music playback and rewind.

**4.46.2.22 - (void) stopEverything**

Stop all effects and bg music.

**4.46.2.23 - OALSimpleAudio: dummy(OALSimpleAudio)**

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (OALSimpleAudio\*) sharedInstance: Get the shared singleton instance.

- (void) purgeSharedInstance: Purge (deallocate) the shared instance.

**4.46.2.24 - (void) unloadAllEffects**

Unload all preloaded effects.

It is useful to put a call to this method in "applicationDidReceiveMemoryWarning" in your app delegate.

**4.46.2.25 - (void) unloadEffect: dummy(NSString\*) filePath**

Unload a preloaded effect.

**Parameters**

<i>filePath</i>	The path containing the sound data that was previously loaded.
-----------------	--

**4.46.3 Member Data Documentation****4.46.3.1 - (ALChannelSource\*) channel [protected]**

The sound channel used by this object.

**4.46.3.2 - (ALContext\*) context [protected]**

The context we are using.

**4.46.3.3 - (ALDevice\*) device [protected]**

The device we are using.

**4.46.3.4 - (uint) pendingLoadCount [protected]**

keeping track of how many effects remain to be loaded

**4.46.3.5 - (NSMutableDictionary\*) preloadCache** [protected]

Cache for preloaded sound samples.

**4.46.4 Property Documentation****4.46.4.1 - (bool) allowIpod** [read, write, assign]

If YES, allow ipod music to continue playing (NOT SUPPORTED ON THE SIMULATOR).

Note: If this is enabled, and another app is playing music, background audio playback will use the SOFTWARE codecs, NOT hardware.

If allowIpod = NO, the application will ALWAYS use hardware decoding.

**See also**

[useHardwareIfAvailable](#)

Default value: YES

**4.46.4.2 - (OALAudioTrack \*) backgroundTrack** [read, assign]

Audio track to play background music.

Background audio track.

**4.46.4.3 - (NSURL \*) backgroundTrackURL** [read, assign]

Background audio URL.

**4.46.4.4 - (bool) bgMuted** [read, write, assign]

Mutes BG music playback.

**4.46.4.5 - (bool) bgPaused** [read, write, assign]

Pauses BG music playback.

**4.46.4.6 - (bool) bgPlaying** [read, assign]

If true, BG music is currently playing.

**4.46.4.7 - (float) bgVolume** [read, write, assign]

Background music playback gain/volume (0.0 - 1.0)

**4.46.4.8 - (bool) effectsMuted** [read, write, assign]

Mutes effects playback.

**4.46.4.9 - (bool) effectsPaused** [read, write, assign]

Pauses effects playback.

**4.46.4.10 - (float) effectsVolume** [read, write, assign]

Master effects gain/volume (0.0 - 1.0)

**4.46.4.11 - (bool) honorSilentSwitch** [read, write, assign]

If true, mute when backgrounded, screen locked, or the ringer switch is turned off (NOT SUPPORTED ON THE SIMULATOR).

Default value: YES

**4.46.4.12 - (bool) interrupted** [read, assign]

If YES, the sound system is interrupted.

**4.46.4.13 - (bool) manuallySuspended** [read, write, assign]

Set to YES to manually suspend the sound system.

**4.46.4.14 - (bool) muted** [read, write, assign]

Mutes all audio.

**4.46.4.15 - (bool) paused** [read, write, assign]

Pauses everything.

**4.46.4.16 - (NSInteger) preloadCacheCount** [read, assign]

The number of items currently in the preload cache.

**4.46.4.17 - (bool) preloadCacheEnabled** [read, write, assign]

Enables/disables the preload cache.

If the preload cache is disabled, effects preloading will do nothing (BG preloading will still work).

**4.46.4.18** - (unsigned int) `reservedSources` [read, write, assign]

The number of sources [OALSimpleAudio](#) is using (max 32 on current iOS devices).

**4.46.4.19** - (bool) `suspended` [read, assign]

If YES, the sound system is suspended.

**4.46.4.20** - (bool) `useHardwareIfAvailable` [read, write, assign]

Determines what to do if no other application is playing audio and `allowIpod` = YES (NOT SUPPORTED ON THE SIMULATOR).

If NO, the application will ALWAYS use software decoding. The advantage to this is that the user can background your application and then start audio playing from another application. If `useHardwareIfAvailable` = YES, the user won't be able to do this.

If this is set to YES, the application will use hardware decoding if no other application is currently playing audio. However, no other application will be able to start playing audio if it wasn't playing already.

Note: This switch has no effect if `allowIpod` = NO.

#### See also

[allowIpod](#)

Default value: YES

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

- `OALSimpleAudio.h`
- `OALSimpleAudio.m`

## 4.47 OALSuspendHandler Class Reference

Provides two controls (`interrupted` and `manuallySuspended`) for suspending a slave object, and also propagates such control messages to interested listeners.

```
#import <OALSuspendHandler.h>
```

### Public Member Functions

- (id) - [initWithTarget:selector:](#)

*Initialize a handler with the specified slave target and selector.*

- (void) - [addSuspendListener](#):  
*Add a listener that will receive manual suspend and interrupt events.*
- (void) - [removeSuspendListener](#):  
*Remove a registered listener.*

## Static Public Member Functions

- ([OALSuspendHandler \\*](#)) + [handlerWithTarget:selector](#):  
*Create a new handler with the specified slave target and selector.*

## Protected Attributes

- [NSMutableArray \\*](#) [listeners](#)  
*Listeners that will receive manualSuspend and interrupt events.*
- [NSMutableArray \\*](#) [manualSuspendStates](#)  
*Holder for the state of manualSuspend in listeners when this object is manually suspended.*
- id [suspendStatusChangeTarget](#)  
*Slave object that is notified when this object suspends or unsuspends.*
- SEL [suspendStatusChangeSelector](#)  
*Selector to be invoked on suspend or unsuspend.*
- bool [manualSuspendLock](#)  
*Holds the current "manually suspended" state.*
- bool [interruptLock](#)  
*Holds the current "interrupted" state.*

## Properties

- bool [manuallySuspended](#)  
*If YES, the manual suspend control is set.*
- bool [interrupted](#)  
*If YES, the interrupt control is set.*



- bool [suspended](#)

*If YES, the slave object is suspended.*

#### 4.47.1 Detailed Description

Provides two controls (interrupted and manuallySuspended) for suspending a slave object, and also propagates such control messages to interested listeners. "interrupted" is meant to be set by the system when an interrupt occurs.

"manuallySuspended" is a user-settable control for suspending an object.

"manuallySuspended" also has an extra step in its processing: When set, the handler makes a note of what its listeners' "manuallySuspended" values are. When cleared, it will only clear a listener's "manuallySuspended" value if it was not set at suspend time. This allows for ad-hoc setting/clearing of "manuallySuspended" in the middle of a handler/listener graph rather than only from the top level.

When either control is set, the slave object will be suspended. When both are cleared, the slave object will be unsuspended.

#### 4.47.2 Member Function Documentation

##### 4.47.2.1 - (void) addSuspendListener: dummy(id<OALSuspendListener>) *listener*

Add a listener that will receive manual suspend and interrupt events.

###### Parameters

<i>listener</i>	The listener to register with this handler.
-----------------	---

##### 4.47.2.2 + (OALSuspendHandler \*) handlerWithTarget: dummy(id) *target* selector:(SEL) *selector*

Create a new handler with the specified slave target and selector.

The selector provided must take a single boolean value like so:

- (void) setSuspended:(bool) value

###### Parameters

<i>target</i>	The slave object that will receive suspend/unsuspend events.
<i>selector</i>	The selector for a "set suspended" method, taking a single boolean parameter.

#### 4.47.2.3 - (id) initWithTarget: dummy(id) target selector:(SEL) selector

Initialize a handler with the specified slave target and selector.

The selector provided must take a single boolean value like so:

- (void) setSuspended:(bool) value

##### Parameters

<i>target</i>	The slave object that will receive suspend/unsuspend events.
<i>selector</i>	The selector for a "set suspended" method, taking a single boolean parameter.

#### 4.47.2.4 - (void) removeSuspendListener: dummy(id<OALSuspendListener>) listener

Remove a registered listener.

##### Parameters

<i>listener</i>	The listener to unregister from this handler.
-----------------	---

### 4.47.3 Member Data Documentation

#### 4.47.3.1 - (bool) interruptLock [protected]

Holds the current "interrupted" state.

#### 4.47.3.2 - (NSMutableArray\*) listeners [protected]

Listeners that will receive manualSuspend and interrupt events.

#### 4.47.3.3 - (bool) manualSuspendLock [protected]

Holds the current "manually suspended" state.

#### 4.47.3.4 - (NSMutableArray\*) manualSuspendStates [protected]

Holder for the state of manualSuspend in listeners when this object is manually suspended.

#### 4.47.3.5 - (SEL) suspendStatusChangeSelector [protected]

Selector to be invoked on suspend or unsuspend.

Takes the signature: setSelected:(bool) value

#### 4.47.3.6 - (id) suspendStatusChangeTarget [protected]

Slave object that is notified when this object suspends or unsuspends.

### 4.47.4 Property Documentation

#### 4.47.4.1 - (bool) interrupted [read, write, assign]

If YES, the interrupt control is set.

#### 4.47.4.2 - (bool) manuallySuspended [read, write, assign]

If YES, the manual suspend control is set.

#### 4.47.4.3 - (bool) suspended [read, assign]

If YES, the slave object is suspended.

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

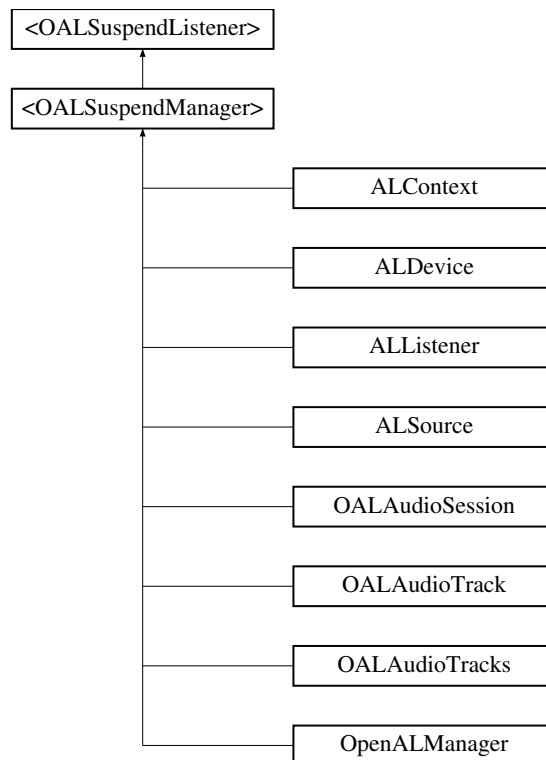
- OALSuspendHandler.h
- OALSuspendHandler.m

## 4.48 <OALSuspendListener> Protocol Reference

Allows an object to participate in interrupt and suspend operations.

```
#import <OALSuspendHandler.h>
```

Inheritance diagram for <OALSuspendListener>:



## Properties

- bool [manuallySuspended](#)  
*Set to YES to manually suspend.*
- bool [interrupted](#)  
*If YES, this object is interrupted.*

### 4.48.1 Detailed Description

Allows an object to participate in interrupt and suspend operations. Objects may hook into **OALAudioSession**'s interrupt and suspend model by calling `[[OALAudioSession sharedInstance] addSuspendListener:self]`.

Note: You must NOT set the "interrupted" property manually. It is designed to be set automatically by system interrupts.

#### See also

[OALAudioSession](#)

## 4.48.2 Property Documentation

### 4.48.2.1 - (bool) interrupted [read, write, assign]

If YES, this object is interrupted.

Note: This property must NOT be set by the user!

Reimplemented in [OALAudioTrack](#).

### 4.48.2.2 - (bool) manuallySuspended [read, write, assign]

Set to YES to manually suspend.

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

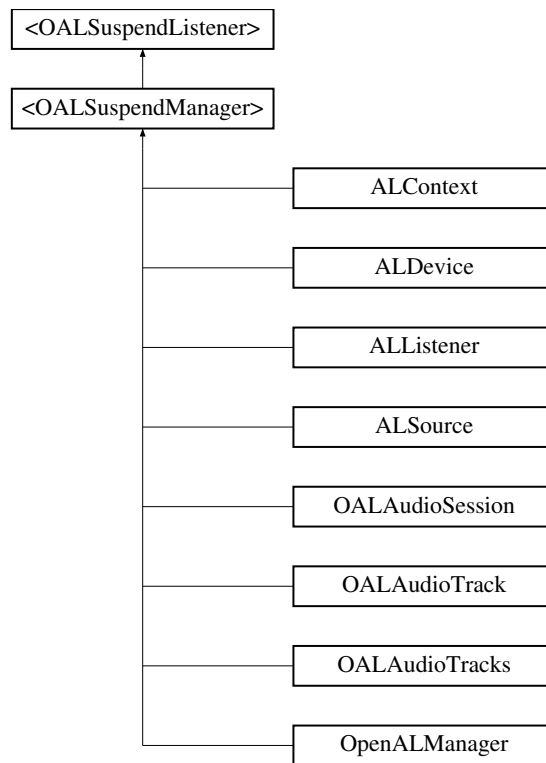
- OALSuspendHandler.h

## 4.49 <OALSuspendManager> Protocol Reference

A suspend manager is a listener that also allows other objects to subscribe to receive events as the manager receives them.

```
#import <OALSuspendHandler.h>
```

Inheritance diagram for <OALSuspendManager>:



## Public Member Functions

- (void) - [addSuspendListener](#):  
*Add a listener that will receive manual suspend and interrupt events.*
- (void) - [removeSuspendListener](#):  
*Remove a registered listener.*

## Properties

- bool [suspended](#)  
*If YES, this object is suspended.*

### 4.49.1 Detailed Description

A suspend manager is a listener that also allows other objects to subscribe to receive events as the manager receives them.

### 4.49.2 Member Function Documentation

#### 4.49.2.1 - (void) addSuspendListener: dummy(id< OALSuspendListener >) listener

Add a listener that will receive manual suspend and interrupt events.

##### Parameters

<i>listener</i>	The listener to register with this handler.
-----------------	---

#### 4.49.2.2 - (void) removeSuspendListener: dummy(id< OALSuspendListener >) listener

Remove a registered listener.

##### Parameters

<i>listener</i>	The listener to unregister from this handler.
-----------------	---

### 4.49.3 Property Documentation

#### 4.49.3.1 - (bool) suspended [read, assign]

If YES, this object is suspended.

Reimplemented in [ALContext](#).

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

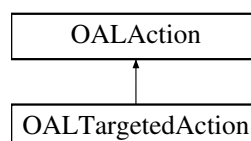
- OALSuspendHandler.h

## 4.50 OALTargetedAction Class Reference

Ignores whatever target it was invoked upon and applies the specified action on the target specified at creation time.

```
#import <OALUtilityActions.h>
```

Inheritance diagram for OALTargetedAction:



## Public Member Functions

- (id) - [initWithTarget:action:](#)  
*Initialize an action.*

## Static Public Member Functions

- (id) + [actionWithTarget:action:](#)  
*Create an action.*

## Protected Attributes

- [OALAction](#) \* [action](#)  
*The action that will be run on the target.*

## Properties

- id [forcedTarget](#)  
*The target which this action will actually be invoked upon.*

### 4.50.1 Detailed Description

Ignores whatever target it was invoked upon and applies the specified action on the target specified at creation time.

### 4.50.2 Member Function Documentation

#### 4.50.2.1 + (id) [actionWithTarget: dummy\(id\) target action:\(OALAction\\*\) action](#)

Create an action.

#### Parameters

<i>target</i>	The target to run the action upon.
<i>action</i>	The action to run.

#### Returns

A new action.



**4.50.2.2 - (id) initWithTarget: dummy(id) target action:(OALAction\*) action**

Initialize an action.

**Parameters**

<i>target</i>	The target to run the action upon.
<i>action</i>	The action to run.

**Returns**

The initialized action.

**4.50.3 Member Data Documentation****4.50.3.1 - (OALAction\*) action [protected]**

The action that will be run on the target.

**4.50.4 Property Documentation****4.50.4.1 - (id) forcedTarget [read, write, assign]**

The target which this action will actually be invoked upon.

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

- OALUtilityActions.h
- OALUtilityActions.m

**4.51 OALTools Class Reference**

Miscellaneous tools used by ObjectAL.

```
#import <OALTools.h>
```

**Static Public Member Functions**

- (NSURL \*) + [urlForPath:](#)  
*Returns the URL corresponding to the specified path.*
- (void) + [notifyExtAudioError:function:description:](#)  
*Notify an error if the specified ExtAudio error code indicates an error.*
- (void) + [notifyAudioSessionError:function:description:](#)  
*Notify an error if the specified AudioSession error code indicates an error.*

### 4.51.1 Detailed Description

Miscellaneous tools used by ObjectAL.

### 4.51.2 Member Function Documentation

#### 4.51.2.1 + (void) notifyAudioSessionError: dummy(OSStatus) *errorCode* function:(const char\*) function description:(NSString\*) *description*, ...

Notify an error if the specified AudioSession error code indicates an error.

This will log the error and also potentially post an audio error notification (OALAudio-ErrorNotification) if it is suspected that this error is a result of the audio session getting corrupted.

##### Parameters

<i>errorCode</i> ,:	The error code returned from an OS call.
<i>function</i> ,:	The function name where the error occurred.
<i>description</i> ,:	A printf-style description of what happened.

#### 4.51.2.2 + (void) notifyExtAudioError: dummy(OSStatus) *errorCode* function:(const char\*) function description:(NSString\*) *description*, ...

Notify an error if the specified ExtAudio error code indicates an error.

This will log the error and also potentially post an audio error notification (OALAudio-ErrorNotification) if it is suspected that this error is a result of the audio session getting corrupted.

##### Parameters

<i>errorCode</i> ,:	The error code returned from an OS call.
<i>function</i> ,:	The function name where the error occurred.
<i>description</i> ,:	A printf-style description of what happened.

#### 4.51.2.3 + (NSURL \*) urlForPath: dummy(NSString\*) *path*

Returns the URL corresponding to the specified path.

If the path is not absolute (starts with a "/"), this method will look for the file in the application's main bundle.

##### Parameters

<i>path</i>	The path to convert to a URL.
-------------	-------------------------------

**Returns**

The corresponding URL or nil if a URL could not be formed.

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

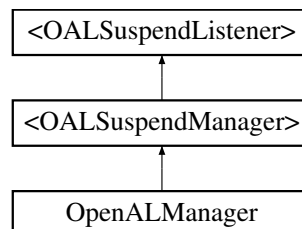
- OALTools.h
- OALTools.m

## 4.52 OpenALManager Class Reference

Manager class for OpenAL objects (ObjectAL).

```
#import <OpenALManager.h>
```

Inheritance diagram for OpenALManager:

**Public Member Functions**

- (void) - [close](#)  
*Close any OS resources in use by this object.*
- (ALBuffer \*) - [bufferFromFile:](#)  
*Load an OpenAL buffer with the contents of an audio file.*
- (ALBuffer \*) - [bufferFromFile:reduceToMono:](#)  
*Load an OpenAL buffer with the contents of an audio file.*
- (ALBuffer \*) - [bufferFromUrl:](#)  
*Load an OpenAL buffer with the contents of an audio file.*
- (ALBuffer \*) - [bufferFromUrl:reduceToMono:](#)  
*Load an OpenAL buffer with the contents of an audio file.*
- (NSString \*) - [bufferAsyncFromFile:target:selector:](#)  
*Load an OpenAL buffer with the contents of an audio file asynchronously.*
- (NSString \*) - [bufferAsyncFromFile:reduceToMono:target:selector:](#)

*Load an OpenAL buffer with the contents of an audio file asynchronously.*

- (NSString \*) - [bufferAsyncFromUrl:target:selector:](#)

*Load an OpenAL buffer with the contents of a URL asynchronously.*

- (NSString \*) - [bufferAsyncFromUrl:reduceToMono:target:selector:](#)

*Load an OpenAL buffer with the contents of a URL asynchronously.*

- (void) - [clearAllBuffers](#)

*Clear all references to sound data from ALL buffers, managed or not.*

- (void) - [notifyDeviceInitializing:](#)

*(INTERNAL USE) Notify that a device is initializing.*

- (void) - [notifyDeviceDeallocating:](#)

*(INTERNAL USE) Notify that a device is deallocating.*

- (void) - [closeOSResources](#)

*(INTERNAL USE) Close any resources belonging to the OS.*

- (void) - [setSuspended:](#)

*(INTERNAL USE) Called by SuspendHandler.*

## Protected Member Functions

- () - [SYNTHESIZE\\_SINGLETON\\_FOR\\_CLASS\\_HEADER](#)

*Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.*

## Protected Attributes

- NSMutableArray \* [devices](#)

*All opened devices.*

- OALSuspendHandler \* [suspendHandler](#)

*Handles suspending and interrupting for this object.*

- NSOperationQueue \* [operationQueue](#)

*Operation queue for asynchronous loading.*

## Properties

- NSArray \* [availableDevices](#)  
*List of available playback devices (NSString\*).*
- NSArray \* [availableCaptureDevices](#)  
*List of available capture devices (NSString\*).*
- ALContext \* [currentContext](#)  
*The current context (some context operations require the context to be the "current" one).*
- NSString \* [defaultCaptureDeviceSpecifier](#)  
*Name of the default capture device.*
- NSString \* [defaultDeviceSpecifier](#)  
*Name of the default playback device.*
- NSArray \* [devices](#)  
*List of all open devices (ALDevice\*).*
- ALdouble [mixerOutputFrequency](#)  
*The frequency of the output mixer.*

### 4.52.1 Detailed Description

Manager class for OpenAL objects (ObjectAL). Keeps track of devices that have been opened, and allows high level OpenAL management.

Provides methods for loading [ALBuffer](#) objects from audio files.

The OpenAL 1.1 specification is available at <http://connect.creativelabs.com/openal/Documentation>

Be sure to read through it (especially the part about distance models) as ObjectAL follows the OpenAL object model.

Alternatively, you may opt to use [OALSimpleAudio](#) for a simpler interface.

### 4.52.2 Member Function Documentation

**4.52.2.1** - (NSString \*) [bufferAsyncFromFile:](#) dummy(NSString\*) *filePath* [reduceToMono:](#)(bool) *reduceToMono* target:(id) *target* selector:(SEL) *selector*

Load an OpenAL buffer with the contents of an audio file asynchronously.

This method will schedule a request to have the buffer created and filled, and then call the specified selector with the newly created buffer.

The buffer's name will be the fully qualified URL of the path.

Returns the fully qualified URL of the path, which you can match up to the buffer name in your callback method.

See the class description note regarding sound file formats.

#### Parameters

<i>filePath</i>	The path of the file containing the audio data.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).
<i>target</i>	The target to call when the buffer is loaded.
<i>selector</i>	The selector to invoke when the buffer is loaded.

#### Returns

The fully qualified URL of the path.

#### 4.52.2.2 - (NSString \*) bufferAsyncFromFile: dummy(NSString\*) filePath target:(id) target selector:(SEL) selector

Load an OpenAL buffer with the contents of an audio file asynchronously.

This method will schedule a request to have the buffer created and filled, and then call the specified selector with the newly created buffer.

The buffer's name will be the fully qualified URL of the path.

Returns the fully qualified URL of the path, which you can match up to the buffer name in your callback method.

See the class description note regarding sound file formats.

#### Parameters

<i>filePath</i>	The path of the file containing the audio data.
<i>target</i>	The target to call when the buffer is loaded.
<i>selector</i>	The selector to invoke when the buffer is loaded.

#### Returns

The fully qualified URL of the path.

#### 4.52.2.3 - (NSString \*) bufferAsyncFromUrl: dummy(NSURL\*) url reduceToMono:(bool) reduceToMono target:(id) target selector:(SEL) selector

Load an OpenAL buffer with the contents of a URL asynchronously.

This method will schedule a request to have the buffer created and filled, and then call the specified selector with the newly created buffer.

The buffer's name will be the fully qualified URL.

Returns the fully qualified URL, which you can match up to the buffer name in your callback method.

See the class description note regarding sound file formats.

#### Parameters

<i>url</i>	The URL of the file containing the audio data.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).
<i>target</i>	The target to call when the buffer is loaded.
<i>selector</i>	The selector to invoke when the buffer is loaded.

#### Returns

The fully qualified URL of the path.

#### 4.52.2.4 - (NSString \*) bufferAsyncFromUrl: dummy(NSURL\*) url target:(id) target selector:(SEL) selector

Load an OpenAL buffer with the contents of a URL asynchronously.

This method will schedule a request to have the buffer created and filled, and then call the specified selector with the newly created buffer.

The buffer's name will be the fully qualified URL.

Returns the fully qualified URL, which you can match up to the buffer name in your callback method.

See the class description note regarding sound file formats.

#### Parameters

<i>url</i>	The URL of the file containing the audio data.
<i>target</i>	The target to call when the buffer is loaded.
<i>selector</i>	The selector to invoke when the buffer is loaded.

#### Returns

The fully qualified URL of the path.

#### 4.52.2.5 - (ALBuffer \*) bufferFromFile: dummy(NSString\*) filePath

Load an OpenAL buffer with the contents of an audio file.

The buffer's name will be the fully qualified URL of the path.

See the class description note regarding sound file formats.

#### Parameters

<i>filePath</i>	The path of the file containing the audio data.
-----------------	---

**Returns**

An [ALBuffer](#) containing the audio data.

#### 4.52.2.6 - (ALBuffer \*) bufferFromFile: dummy(NSString\*) filePath reduceToMono:(bool) *reduceToMono*

Load an OpenAL buffer with the contents of an audio file.

The buffer's name will be the fully qualified URL of the path.

See the class description note regarding sound file formats.

**Parameters**

<i>filePath</i>	The path of the file containing the audio data.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).

**Returns**

An [ALBuffer](#) containing the audio data.

#### 4.52.2.7 - (ALBuffer \*) bufferFromUrl: dummy(NSURL\*) url

Load an OpenAL buffer with the contents of an audio file.

The buffer's name will be the fully qualified URL.

See the class description note regarding sound file formats.

**Parameters**

<i>url</i>	The URL of the file containing the audio data.
------------	--

**Returns**

An [ALBuffer](#) containing the audio data.

#### 4.52.2.8 - (ALBuffer \*) bufferFromUrl: dummy(NSURL\*) url reduceToMono:(bool) *reduceToMono*

Load an OpenAL buffer with the contents of an audio file.

The buffer's name will be the fully qualified URL.

See the class description note regarding sound file formats.

**Parameters**

<i>url</i>	The URL of the file containing the audio data.
<i>reduce-ToMono</i>	If true, reduce the sample to mono (stereo samples don't support panning or positional audio).



**Returns**

An [ALBuffer](#) containing the audio data.

**4.52.2.9 - (void) clearAllBuffers**

Clear all references to sound data from ALL buffers, managed or not.

**4.52.2.10 - (void) close**

Close any OS resources in use by this object.

Any operations called on this object after closing will likely fail.

**4.52.2.11 - (void) closeOSResources**

(INTERNAL USE) Close any resources belonging to the OS.

**4.52.2.12 - (void) notifyDeviceDeallocating: dummy(ALDevice\*) device**

(INTERNAL USE) Notify that a device is deallocating.

**4.52.2.13 - (void) notifyDeviceInitializing: dummy(ALDevice\*) device**

(INTERNAL USE) Notify that a device is initializing.

**4.52.2.14 - (void) setSuspended: dummy(bool) value**

(INTERNAL USE) Called by SuspendHandler.

**4.52.2.15 - OpenALManager: dummy(OpenALManager)**

Singleton implementation providing "sharedInstance" and "purgeSharedInstance" methods.

- (**OpenALManager\***) **sharedInstance**: Get the shared singleton instance.

- (**void**) **purgeSharedInstance**: Purge (deallocate) the shared instance.

**4.52.3 Member Data Documentation****4.52.3.1 - (NSMutableArray\*) devices** [protected]

All opened devices.

**4.52.3.2 - (NSOperationQueue\*) operationQueue [protected]**

Operation queue for asynchronous loading.

**4.52.3.3 - (OALSuspendHandler\*) suspendHandler [protected]**

Handles suspending and interrupting for this object.

**4.52.4 Property Documentation****4.52.4.1 - (NSArray \*) availableCaptureDevices [read, assign]**

List of available capture devices (NSString\*).

**4.52.4.2 - (NSArray \*) availableDevices [read, assign]**

List of available playback devices (NSString\*).

**4.52.4.3 - (ALContext \*) currentContext [read, write, assign]**

The current context (some context operations require the context to be the "current" one).

**4.52.4.4 - (NSString \*) defaultCaptureDeviceSpecifier [read, assign]**

Name of the default capture device.

**4.52.4.5 - (NSString \*) defaultDeviceSpecifier [read, assign]**

Name of the default playback device.

**4.52.4.6 - (NSArray\*) devices [read, assign]**

List of all open devices (ALDevice\*).

**4.52.4.7 - (ALdouble) mixerOutputFrequency [read, write, assign]**

The frequency of the output mixer.

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

- OpenALManager.h
- OpenALManager.m

# Index

- abortPlaybackResume
  - ALSource, [71](#)
- action
  - OALTargetedAction, [213](#)
- actionIndex
  - OALSequentialActions, [189](#)
- actions
  - OALConcurrentActions, [164](#)
  - OALSequentialActions, [189](#)
- actions:
  - OALConcurrentActions, [163](#)
  - OALSequentialActions, [188](#)
- actionsFromArray:
  - OALConcurrentActions, [163](#)
  - OALSequentialActions, [188](#)
- actionsToAdd
  - OALActionManager, [128](#)
- actionsToRemove
  - OALActionManager, [128](#)
- actionsWithDuration
  - OALConcurrentActions, [164](#)
- actionWithCallTarget:selector:
  - OALCallAction, [159](#)
- actionWithCallTarget:selector:withObject:
  - OALCallAction, [160](#)
- actionWithCallTarget:selector:withObject:withObject:
  - OALCallAction, [160](#)
- actionWithDuration:delta:
  - OALMoveByAction, [176](#)
- actionWithDuration:endValue:
  - OALFunctionAction, [168](#)
- actionWithDuration:endValue:function:
  - OALFunctionAction, [169](#)
- actionWithDuration:position:
  - OALMoveToAction, [179](#)
- actionWithDuration:startValue:endValue:function:
  - OALFunctionAction, [169](#)
- actionWithPosition:
  - OALPlaceAction, [183](#)
- actionWithTarget:action:
  - OALTargetedAction, [212](#)
- actionWithUnitsPerSecond:delta:
  - OALMoveByAction, [176](#)
- actionWithUnitsPerSecond:position:
  - OALMoveToAction, [179](#)
- addSource:
  - ALSoundSourcePool, [65](#)
- addSuspendListener:
  - OALSuspendHandler, [205](#)
  - OALSuspendManager-p, [211](#)
- ALBuffer, [21](#)
  - bits, [23](#)
  - bufferData, [23](#)
  - bufferId, [23](#)
  - bufferWithName:data:size:format:frequency:, [22](#)
  - channels, [23](#)
  - device, [24](#)
  - duration, [24](#)
  - format, [24](#)
  - freeDataOnDestroy, [24](#)
  - frequency, [24](#)
  - initWithName:data:size:format:frequency:, [23](#)
  - name, [24](#)
  - size, [24](#)
- ALCaptureDevice, [24](#)
  - captureSamples, [28](#)
  - close, [26](#)
  - closeOSResources, [26](#)
  - device, [28](#)
  - deviceWithDeviceSpecifier:frequency:format:bufferSize:, [26](#)
  - extensions, [28](#)
  - getProcAddress:, [26](#)
  - initWithDeviceSpecifier:frequency:format:bufferSize:, [27](#)
  - isExtensionPresent:, [27](#)
  - majorVersion, [28](#)
  - minorVersion, [28](#)
  - moveSamples:toBuffer:, [27](#)
  - startCapture, [28](#)

- stopCapture, 28
- ALChannelSource, 29
  - channelWithSources:, 32
  - closeOSResources, 32
  - coneInnerAngle, 33
  - coneOuterAngle, 33
  - coneOuterGain, 33
  - context, 37
  - currentFadeCallbackCount, 34
  - currentPanCallbackCount, 34
  - currentPitchCallbackCount, 34
  - direction, 34
  - expectedFadeCallbackCount, 34
  - expectedPanCallbackCount, 34
  - expectedPitchCallbackCount, 34
  - fadeCompleteSelector, 34
  - fadeCompleteTarget, 34
  - gain, 34
  - initWithSources:, 33
  - interruptible, 35
  - looping, 35
  - maxDistance, 35
  - maxGain, 35
  - minGain, 35
  - muted, 35
  - onFadeComplete:, 33
  - onPanComplete:, 33
  - onPitchComplete:, 33
  - panCompleteSelector, 35
  - panCompleteTarget, 35
  - paused, 36
  - pitch, 36
  - pitchCompleteSelector, 36
  - pitchCompleteTarget, 36
  - position, 36
  - referenceDistance, 36
  - reservedSources, 37
  - resetToDefault, 33
  - rolloffFactor, 36
  - sourcePool, 37
  - sourceRelative, 36
  - sourceType, 36
  - velocity, 37
- ALContext, 37
  - alVersion, 44
  - attributes, 44
  - clearBuffers, 40
  - close, 40
  - closeOSResources, 40
  - context, 44
  - contextOnDevice:attributes:, 41
  - contextOnDevice:outputFrequency:refreshIntervals:synchronousCo
    - 41
  - device, 45
  - distanceModel, 45
  - dopplerFactor, 45
  - ensureContextIsCurrent, 41
  - extensions, 45
  - getProcAddress:, 42
  - initWithDevice:attributes:, 42
  - initWithDevice:outputFrequency:refreshIntervals:synchronousCo
    - 42
  - isExtensionPresent:, 43
  - listener, 45
  - notifySourceDeallocating:, 43
  - notifySourceInitializing:, 43
  - process, 43
  - renderer, 45
  - setSuspended:, 43
  - sources, 44, 45
  - speedOfSound, 45
  - stopAllSounds, 44
  - suspended, 44
  - suspendHandler, 44
  - vendor, 46
- ALDevice, 46
  - clearBuffers, 48
  - close, 48
  - closeOSResources, 48
  - contexts, 49, 50
  - device, 50
  - deviceWithDeviceSpecifier:, 48
  - extensions, 50
  - getProcAddress:, 48
  - initWithDeviceSpecifier:, 48
  - isExtensionPresent:, 49
  - majorVersion, 50
  - minorVersion, 50
  - notifyContextDeallocating:, 49
  - notifyContextInitializing:, 49
  - suspendHandler, 49
- ALListener, 50
  - context, 53
  - gain, 53
  - initWithContext:, 52
  - listenerForContext:, 52
  - muted, 53
  - orientation, 53
  - position, 53
  - suspendHandler, 52

- velocity, 53
- allowIpad
  - OALAudioSession, 138
  - OALSimpleAudio, 201
- ALOrientation, 53
  - at, 54
  - up, 54
- ALPoint, 54
  - x, 55
  - y, 55
  - z, 55
- ALSoundSource-p, 55
  - clear, 58
  - close, 58
  - coneInnerAngle, 61
  - coneOuterAngle, 61
  - coneOuterGain, 61
  - direction, 61
  - fadeTo:duration:target:selector:, 58
  - gain, 61
  - interruptible, 61
  - looping, 61
  - maxDistance, 62
  - maxGain, 62
  - minGain, 62
  - muted, 62
  - pan, 62
  - panTo:duration:target:selector:, 59
  - paused, 62
  - pitch, 62
  - pitchTo:duration:target:selector:, 59
  - play:, 59
  - play:gain:pitch:pan:loop:, 59
  - play:loop:, 60
  - playing, 62
  - position, 63
  - referenceDistance, 63
  - rewind, 60
  - rolloffFactor, 63
  - sourceRelative, 63
  - sourceType, 63
  - stop, 60
  - stopActions, 60
  - stopFade, 60
  - stopPan, 60
  - stopPitch, 61
  - velocity, 63
  - volume, 63
- ALSoundSourcePool, 64
  - addSource:, 65
  - close, 65
  - closeOSResources, 65
  - getFreeSource:, 65
  - moveToHead:, 65
  - pool, 66
  - removeSource:, 66
  - sources, 66
- ALSource, 66
  - abortPlaybackResume, 71
  - buffer, 72
  - buffersProcessed, 72
  - buffersQueued, 72
  - closeOSResources, 69
  - context, 72
  - delayedResumePlayback, 69
  - gain, 71
  - gainAction, 71
  - initOnContext:, 69
  - interruptible, 71
  - muted, 72
  - offsetInBytes, 73
  - offsetInSamples, 73
  - offsetInSeconds, 73
  - panAction, 72
  - pitchAction, 72
  - play, 70
  - queueBuffer:, 70
  - queueBuffers:, 70
  - setSuspended:, 70
  - shadowState, 72
  - source, 70
  - sourceId, 73
  - sourceOnContext:, 70
  - state, 73
  - suspendHandler, 72
  - unqueueBuffer:, 71
  - unqueueBuffers:, 71
- ALVector, 73
  - x, 74
  - y, 74
  - z, 74
- alVersion
  - ALContext, 44
- ALWrapper, 74
  - buffer3f:parameter:v1:v2:v3:, 82
  - buffer3i:parameter:v1:v2:v3:, 82
  - bufferData:format:data:size:frequency:, 82
  - bufferDataStatic:format:data:size:frequency:, 83

- bufferf:parameter:value:, 83
- bufferfv:parameter:values:, 83
- bufferi:parameter:value:, 84
- bufferiv:parameter:values:, 84
- captureSamples:buffer:numSamples:, 84
- checkIfSuccessful, 85
- checkIfSuccessfulWithDevice, 85
- closeCaptureDevice:, 85
- closeDevice:, 85
- createContext:attributes:, 86
- decodeNullSeparatedStringList:, 86
- decodeSpaceSeparatedStringList:, 86
- deleteBuffer:, 86
- deleteBuffers:numBuffers:, 87
- deleteSource:, 87
- deleteSources:numSources:, 87
- destroyContext:, 88
- disable:, 88
- distanceModel:, 88
- dopplerFactor:, 88
- enable:, 89
- genBuffer, 89
- genBuffers:numBuffers:, 89
- genSource, 89
- genSources:numSources:, 89
- getBoolean:, 90
- getBooleanv:values:, 90
- getBuffer3f:parameter:v1:v2:v3:, 90
- getBuffer3i:parameter:v1:v2:v3:, 91
- getBufferf:parameter:, 91
- getBufferfv:parameter:values:, 91
- getBufferi:parameter:, 92
- getBufferiv:parameter:values:, 92
- getContextsDevice:, 92
- getContextsDevice:deviceReference:, 92
- getCurrentContext, 93
- getDouble:, 93
- getDoublev:values:, 93
- getEnumValue:, 93
- getEnumValue:name:, 94
- getFloat:, 94
- getFloatv:values:, 94
- getInteger:, 95
- getInteger:attribute:, 95
- getInteger:attribute:size:data:, 95
- getInteger:values:, 95
- getListener3f:v1:v2:v3:, 96
- getListener3i:v1:v2:v3:, 96
- getListenerf:, 96
- getListenerfv:values:, 97
- getListeneri:, 97
- getListeneriv:values:, 97
- getMixerOutputDataRate, 97
- getNullSeparatedStringList:, 98
- getNullSeparatedStringList:attribute:, 98
- getProcAddress:, 98
- getProcAddress:name:, 99
- getSource3f:parameter:v1:v2:v3:, 99
- getSource3i:parameter:v1:v2:v3:, 99
- getSourcef:parameter:, 100
- getSourcefv:parameter:values:, 100
- getSourcei:parameter:, 100
- getSourceiv:parameter:values:, 100
- getSpaceSeparatedStringList:, 101
- getSpaceSeparatedStringList:attribute:, 101
- getString:, 101
- getString:attribute:, 102
- isBuffer:, 102
- isEnabled:, 102
- isExtensionPresent:, 102
- isExtensionPresent:name:, 103
- isSource:, 103
- listener3f:v1:v2:v3:, 103
- listener3i:v1:v2:v3:, 104
- listenerf:value:, 104
- listenerfv:values:, 104
- listeneri:value:, 104
- listeneriv:values:, 105
- makeContextCurrent:, 105
- makeContextCurrent:deviceReference:, 105
- openCaptureDevice:frequency:format:bufferSize:, 106
- openDevice:, 106
- processContext:, 106
- setMixerOutputDataRate:, 106
- source3f:parameter:v1:v2:v3:, 107
- source3i:parameter:v1:v2:v3:, 107
- sourcef:parameter:value:, 107
- sourcefv:parameter:values:, 108
- sourcei:parameter:value:, 108
- sourceiv:parameter:values:, 108
- sourcePause:, 108
- sourcePausev:numSources:, 109
- sourcePlay:, 109
- sourcePlayv:numSources:, 109

- sourceQueueBuffers:numBuffers:bufferIds:, OALSimpleAudio, 201
- 110
- sourceRewind:, 110
- sourceRewindv:numSources:, 110
- sourceStop:, 110
- sourceStopv:numSources:, 111
- sourceUnqueueBuffers:numBuffers:bufferIds:ALWrapper, 82
- 111
- speedOfSound:, 111
- startCapture:, 111
- stopCapture:, 112
- suspendContext:, 112
- at
  - ALOrientation, 54
- attributes
  - ALContext, 44
- audioDataWithStartFrame:numFrames:bufferSize:OpenALManager, 219
- OALAudioFile, 130
- audioRoute
  - OALAudioSession, 138
- audioSessionActive
  - OALAudioSession, 138
- audioSessionCategory
  - OALAudioSession, 138
- audioSessionDelegate
  - OALAudioSession, 139
- audioSessionWasActive
  - OALAudioSession, 138
- audioTrack
  - OAL\_AsyncAudioTrackOperation, 117
- autoPreload
  - OALAudioTrack, 153
- availableCaptureDevices
  - OpenALManager, 222
- availableDevices
  - OpenALManager, 222
- averagePowerForChannel:
  - OALAudioTrack, 145
- backgroundTrack
  - OALSimpleAudio, 201
- backgroundTrackURL
  - OALSimpleAudio, 201
- bgMuted
  - OALSimpleAudio, 201
- bgPaused
  - OALSimpleAudio, 201
- bgPlaying
  - OALSimpleAudio, 201
- bgVolume
  - ALBuffer, 23
  - bits
    - ALBuffer, 23
  - buffer
    - ALSource, 72
  - buffer3f:parameter:v1:v2:v3:
    - ALWrapper, 82
  - buffer3i:parameter:v1:v2:v3:
    - ALWrapper, 82
  - bufferAsyncFromFile:reduceToMono:target:selector:
    - OpenALManager, 217
  - bufferAsyncFromFile:target:selector:
    - OpenALManager, 218
  - bufferAsyncFromUrl:reduceToMono:target:selector:
    - OpenALManager, 218
  - bufferAsyncFromUrl:target:selector:
    - OpenALManager, 219
  - bufferData
    - ALBuffer, 23
  - bufferData:format:data:size:frequency:
    - ALWrapper, 82
  - bufferDataStatic:format:data:size:frequency:
    - ALWrapper, 83
  - bufferf:parameter:value:
    - ALWrapper, 83
  - bufferFromFile:
    - OpenALManager, 219
  - bufferFromFile:reduceToMono:
    - OpenALManager, 220
  - bufferFromUrl:
    - OpenALManager, 220
  - bufferFromUrl:reduceToMono:
    - OALAudioFile, 130
    - OpenALManager, 220
  - bufferfv:parameter:values:
    - ALWrapper, 83
  - bufferi:parameter:value:
    - ALWrapper, 84
  - bufferId
    - ALBuffer, 23
  - bufferiv:parameter:values:
    - ALWrapper, 84
  - bufferNamed:startFrame:numFrames:
    - OALAudioFile, 131
  - buffersProcessed
    - ALSource, 72
  - buffersQueued
    - ALSource, 72
  - bufferWithName:data:size:format:frequency:
    - ALBuffer, 22

- callTarget
  - OALCallAction, [161](#)
- captureSamples
  - ALCaptureDevice, [28](#)
- captureSamples:buffer:numSamples:
  - ALWrapper, [84](#)
- channel
  - OALSimpleAudio, [200](#)
- channels
  - ALBuffer, [23](#)
- channelWithSources:
  - ALChannelSource, [32](#)
- checkIfSuccessful
  - ALWrapper, [85](#)
- checkIfSuccessfulWithDevice
  - ALWrapper, [85](#)
- clear
  - ALSoundSource-p, [58](#)
  - OALAudioTrack, [145](#)
- clearAllBuffers
  - OpenALManager, [221](#)
- clearBuffers
  - ALContext, [40](#)
  - ALDevice, [48](#)
- close
  - ALCaptureDevice, [26](#)
  - ALContext, [40](#)
  - ALDevice, [48](#)
  - ALSoundSource-p, [58](#)
  - ALSoundSourcePool, [65](#)
  - OALAudioFile, [131](#)
  - OALAudioSession, [135](#)
  - OALAudioTrack, [145](#)
  - OALAudioTracks, [157](#)
  - OALSimpleAudio, [194](#)
  - OpenALManager, [221](#)
- closeCaptureDevice:
  - ALWrapper, [85](#)
- closeDevice:
  - ALWrapper, [85](#)
- closeOSResources
  - ALCaptureDevice, [26](#)
  - ALChannelSource, [32](#)
  - ALContext, [40](#)
  - ALDevice, [48](#)
  - ALSoundSourcePool, [65](#)
  - ALSource, [69](#)
  - OALAudioFile, [131](#)
  - OALAudioSession, [135](#)
  - OALAudioTrack, [145](#)
  - OALAudioTracks, [157](#)
  - OALSimpleAudio, [194](#)
  - OpenALManager, [221](#)
- coneInnerAngle
  - ALChannelSource, [33](#)
  - ALSoundSource-p, [61](#)
- coneOuterAngle
  - ALChannelSource, [33](#)
  - ALSoundSource-p, [61](#)
- coneOuterGain
  - ALChannelSource, [33](#)
  - ALSoundSource-p, [61](#)
- context
  - ALChannelSource, [37](#)
  - ALContext, [44](#)
  - ALListener, [53](#)
  - ALSource, [72](#)
  - OALSimpleAudio, [200](#)
- contextOnDevice:attributes:
  - ALContext, [41](#)
- contextOnDevice:outputFrequency:refreshIntervals:synchronousContext:
  - ALContext, [41](#)
- contexts
  - ALDevice, [49, 50](#)
- createContext:attributes:
  - ALWrapper, [86](#)
- currentAction
  - OALSequentialActions, [189](#)
- currentContext
  - OpenALManager, [222](#)
- currentFadeCallbackCount
  - ALChannelSource, [34](#)
- currentlyLoadedUrl
  - OALAudioTrack, [153](#)
- currentPanCallbackCount
  - ALChannelSource, [34](#)
- currentPitchCallbackCount
  - ALChannelSource, [34](#)
- currentTime
  - OALAudioTrack, [153](#)
- decodeNullSeparatedStringList:
  - ALWrapper, [86](#)
- decodeSpaceSeparatedStringList:
  - ALWrapper, [86](#)
- defaultCaptureDeviceSpecifier
  - OpenALManager, [222](#)
- defaultDeviceSpecifier
  - OpenALManager, [222](#)
- defaultFunction



- OALFunctionAction, [169](#)
- delayedResumePlayback
  - ALSource, [69](#)
- delegate
  - OALAudioTrack, [153](#)
- deleteBuffer:
  - ALWrapper, [86](#)
- deleteBuffers:numBuffers:
  - ALWrapper, [87](#)
- deleteSource:
  - ALWrapper, [87](#)
- deleteSources:numSources:
  - ALWrapper, [87](#)
- delta
  - OALFunctionAction, [171](#)
  - OALMoveByAction, [177](#)
  - OALMoveToAction, [180](#)
- destroyContext:
  - ALWrapper, [88](#)
- device
  - ALBuffer, [24](#)
  - ALCaptureDevice, [28](#)
  - ALContext, [45](#)
  - ALDevice, [50](#)
  - OALSimpleAudio, [200](#)
- deviceCurrentTime
  - OALAudioTrack, [153](#)
- devices
  - OpenALManager, [221](#), [222](#)
- deviceWithDeviceSpecifier:
  - ALDevice, [48](#)
- deviceWithDeviceSpecifier:frequency:format:bufferSize:
  - ALCaptureDevice, [26](#)
- direction
  - ALChannelSource, [34](#)
  - ALSoundSource-p, [61](#)
- disable:
  - ALWrapper, [88](#)
- distanceModel
  - ALContext, [45](#)
- distanceModel:
  - ALWrapper, [88](#)
- dopplerFactor
  - ALContext, [45](#)
- dopplerFactor:
  - ALWrapper, [88](#)
- doResetTimeDelta:
  - OALActionManager, [127](#)
- duration
  - ALBuffer, [24](#)
- OALAction, [125](#)
- OALAudioTrack, [154](#)
- effectsMuted
  - OALSimpleAudio, [201](#)
- effectsPaused
  - OALSimpleAudio, [202](#)
- effectsVolume
  - OALSimpleAudio, [202](#)
- elapsed
  - OALAction, [125](#)
- enable:
  - ALWrapper, [89](#)
- endValue
  - OALFunctionAction, [171](#)
- ensureContextIsCurrent
  - ALContext, [41](#)
- expectedFadeCallbackCount
  - ALChannelSource, [34](#)
- expectedPanCallbackCount
  - ALChannelSource, [34](#)
- expectedPitchCallbackCount
  - ALChannelSource, [34](#)
- extensions
  - ALCaptureDevice, [28](#)
  - ALContext, [45](#)
  - ALDevice, [50](#)
- fadeCompleteSelector
  - ALChannelSource, [34](#)
- fadeCompleteTarget
  - ALChannelSource, [34](#)
- fadeTo:duration:target:selector:
  - ALSoundSource-p, [58](#)
  - OALAudioTrack, [145](#)
- fileHandle
  - OALAudioFile, [132](#)
- fileWithURL:reduceToMono:
  - OALAudioFile, [131](#)
- forcedTarget
  - OALTargetedAction, [213](#)
- forceEndInterruption
  - OALAudioSession, [136](#)
- format
  - ALBuffer, [24](#)
- freeDataOnDestroy
  - ALBuffer, [24](#)
- frequency
  - ALBuffer, [24](#)
- function

- OALExponentialFunction, [165](#)
- OALFunctionAction, [171](#)
- OALLinearFunction, [173](#)
- OALLogarithmicFunction, [174](#)
- OALReverseFunction, [185](#)
- OALSCurveFunction, [186](#)
- functionWithFunction:
  - OALReverseFunction, [184](#)
- gain
  - ALChannelSource, [34](#)
  - ALListener, [53](#)
  - ALSoundSource-p, [61](#)
  - ALSource, [71](#)
  - OAL\_GainProtocol-p, [120](#)
  - OALAudioTrack, [154](#)
- gainAction
  - ALSource, [71](#)
  - OALAudioTrack, [152](#)
- genBuffer
  - ALWrapper, [89](#)
- genBuffers:numBuffers:
  - ALWrapper, [89](#)
- genSource
  - ALWrapper, [89](#)
- genSources:numSources:
  - ALWrapper, [89](#)
- getBoolean:
  - ALWrapper, [90](#)
- getBooleanv:values:
  - ALWrapper, [90](#)
- getBuffer3f:parameter:v1:v2:v3:
  - ALWrapper, [90](#)
- getBuffer3i:parameter:v1:v2:v3:
  - ALWrapper, [91](#)
- getBufferf:parameter:
  - ALWrapper, [91](#)
- getBufferfv:parameter:values:
  - ALWrapper, [91](#)
- getBufferi:parameter:
  - ALWrapper, [92](#)
- getBufferiv:parameter:values:
  - ALWrapper, [92](#)
- getContextsDevice:
  - ALWrapper, [92](#)
- getContextsDevice:deviceReference:
  - ALWrapper, [92](#)
- getCurrentContext
  - ALWrapper, [93](#)
- getDouble:
  - ALWrapper, [93](#)
- getDoublev:values:
  - ALWrapper, [93](#)
- getEnumValue:
  - ALWrapper, [93](#)
- getEnumValue:name:
  - ALWrapper, [94](#)
- getFloat:
  - ALWrapper, [94](#)
- getFloatProperty:
  - OALAudioSession, [136](#)
- getFloatv:values:
  - ALWrapper, [94](#)
- getFreeSource:
  - ALSoundSourcePool, [65](#)
- getInteger:
  - ALWrapper, [95](#)
- getInteger:attribute:
  - ALWrapper, [95](#)
- getIntegeriv:attribute:size:data:
  - ALWrapper, [95](#)
- getIntegeriv:values:
  - ALWrapper, [95](#)
- getIntProperty:
  - OALAudioSession, [136](#)
- getListener3f:v1:v2:v3:
  - ALWrapper, [96](#)
- getListener3i:v1:v2:v3:
  - ALWrapper, [96](#)
- getListenerf:
  - ALWrapper, [96](#)
- getListenerfv:values:
  - ALWrapper, [97](#)
- getListeneri:
  - ALWrapper, [97](#)
- getListeneriv:values:
  - ALWrapper, [97](#)
- getMixerOutputDataRate
  - ALWrapper, [97](#)
- getNullSeparatedStringList:
  - ALWrapper, [98](#)
- getNullSeparatedStringList:attribute:
  - ALWrapper, [98](#)
- getProcAddress:
  - ALCaptureDevice, [26](#)
  - ALContext, [42](#)
  - ALDevice, [48](#)
  - ALWrapper, [98](#)
- getProcAddress:name:
  - ALWrapper, [99](#)

- getSource3f:parameter:v1:v2:v3:
  - ALWrapper, [99](#)
- getSource3i:parameter:v1:v2:v3:
  - ALWrapper, [99](#)
- getSourcef:parameter:
  - ALWrapper, [100](#)
- getSourcefv:parameter:values:
  - ALWrapper, [100](#)
- getSourcei:parameter:
  - ALWrapper, [100](#)
- getSourceiv:parameter:values:
  - ALWrapper, [100](#)
- getSpaceSeparatedStringList:
  - ALWrapper, [101](#)
- getSpaceSeparatedStringList:attribute:
  - ALWrapper, [101](#)
- getString:
  - ALWrapper, [101](#)
- getString:attribute:
  - ALWrapper, [102](#)
- getStringProperty:
  - OALAudioSession, [136](#)
- handleInterruptions
  - OALAudioSession, [139](#)
- handlerWithTarget:selector:
  - OALSuspendHandler, [205](#)
- hardwareMuted
  - OALAudioSession, [139](#)
- hardwareVolume
  - OALAudioSession, [139](#)
- honorSilentSwitch
  - OALAudioSession, [139](#)
  - OALSimpleAudio, [202](#)
- initOnContext:
  - ALSource, [69](#)
- initOnDevice:attributes:
  - ALContext, [42](#)
- initOnDevice:outputFrequency:refreshIntervals:synchronous:
  - ALContext, [42](#)
- initWithActions:
  - OALConcurrentActions, [164](#)
  - OALSequentialActions, [188](#)
- initWithCallTarget:selector:
  - OALCallAction, [160](#)
- initWithCallTarget:selector:withObject:
  - OALCallAction, [160](#)
- initWithCallTarget:selector:withObject:withObject:
  - OALCallAction, [161](#)
- initWithContext:
  - ALListener, [52](#)
- initWithDeviceSpecifier:
  - ALDevice, [48](#)
- initWithDeviceSpecifier:frequency:format:bufferSize:
  - ALCaptureDevice, [27](#)
- initWithDuration:
  - OALAction, [124](#)
- initWithDuration:delta:
  - OALMoveByAction, [176](#)
- initWithDuration:endValue:
  - OALFunctionAction, [169](#)
- initWithDuration:endValue:function:
  - OALFunctionAction, [170](#)
- initWithDuration:position:
  - OALMoveToAction, [179](#)
- initWithDuration:startValue:endValue:function:
  - OALFunctionAction, [170](#)
- initWithFunction:
  - OALReverseFunction, [185](#)
- initWithName:data:size:format:frequency:
  - ALBuffer, [23](#)
- initWithPosition:
  - OALPlaceAction, [183](#)
- initWithSources:
  - ALChannelSource, [33](#)
  - OALSimpleAudio, [194](#)
- initWithTarget:action:
  - OALTargetedAction, [212](#)
- initWithTarget:selector:
  - OALSuspendHandler, [205](#)
- initWithTrack:url:loops:target:selector:
  - OAL\_AsyncAudioTrackPlayOperation, [119](#)
- initWithTrack:url:seekTime:target:selector:
  - OAL\_AsyncAudioTrackOperation, [117](#)
- initWithUnitsPerSecond:delta:
  - OALMoveByAction, [177](#)
- initWithUnitsPerSecond:position:
  - OALMoveToAction, [180](#)
- initWithUrl:reduceToMono:
  - OALAudioFile, [131](#)
- initWithUrl:reduceToMono:target:selector:
  - OAL\_AsyncALBufferLoadOperation, [114](#)
- internalPreloadEffect:reduceToMono:
  - OALSimpleAudio, [194](#)
- interrupted
  - OALAudioTrack, [152](#)
  - OALSimpleAudio, [202](#)
  - OALSuspendHandler, [207](#)

- OALSuspendListener-p, 209
- interruptible
  - ALChannelSource, 35
  - ALSoundSource-p, 61
  - ALSource, 71
- interruptLock
  - OALSuspendHandler, 206
- IOSVersion, 112
  - SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_OALFunctionAction, 171
  - HEADER, 113
  - version, 113
- ipodDucking
  - OALAudioSession, 139
- ipodPlaying
  - OALAudioSession, 139
- isBuffer:
  - ALWrapper, 102
- isEnabled:
  - ALWrapper, 102
- isExtensionPresent:
  - ALCaptureDevice, 27
  - ALContext, 43
  - ALDevice, 49
  - ALWrapper, 102
- isExtensionPresent:name:
  - ALWrapper, 103
- isSource:
  - ALWrapper, 103
- lastResetTime
  - OALAudioSession, 138
- lastTimestamp
  - OALActionManager, 128
- listener
  - ALContext, 45
- listener3f:v1:v2:v3:
  - ALWrapper, 103
- listener3i:v1:v2:v3:
  - ALWrapper, 104
- listenerf:value:
  - ALWrapper, 104
- listenerForContext:
  - ALListener, 52
- listenerfv:values:
  - ALWrapper, 104
- listeneri:value:
  - ALWrapper, 104
- listeneriv:values:
  - ALWrapper, 105
- listeners
  - OALSuspendHandler, 206
- looping
  - ALChannelSource, 35
  - ALSoundSource-p, 61
- loops
  - OAL\_AsyncAudioTrackPlayOperation, 119
- lowValue
  - majorVersion
    - ALCaptureDevice, 28
    - ALDevice, 50
  - makeContextCurrent:
    - ALWrapper, 105
  - makeContextCurrent:deviceReference:
    - ALWrapper, 105
  - manuallySuspended
    - OALSimpleAudio, 202
    - OALSuspendHandler, 207
    - OALSuspendListener-p, 209
  - manualSuspendLock
    - OALSuspendHandler, 206
  - manualSuspendStates
    - OALSuspendHandler, 206
  - maxDistance
    - ALChannelSource, 35
    - ALSoundSource-p, 62
  - maxGain
    - ALChannelSource, 35
    - ALSoundSource-p, 62
  - meteringEnabled
    - OALAudioTrack, 154
  - minGain
    - ALChannelSource, 35
    - ALSoundSource-p, 62
  - minorVersion
    - ALCaptureDevice, 28
    - ALDevice, 50
  - mixerOutputFrequency
    - OpenALManager, 222
  - moveSamples:toBuffer:
    - ALCaptureDevice, 27
  - moveToHead:
    - ALSoundSourcePool, 65
  - muted
    - ALChannelSource, 35
    - ALListener, 53
    - ALSoundSource-p, 62
    - ALSource, 72

- OALAudioTrack, 154
- OALAudioTracks, 158
- OALSimpleAudio, 202
- name
  - ALBuffer, 24
- notifyActionStarted:
  - OALActionManager, 127
- notifyActionStopped:
  - OALActionManager, 127
- notifyAudioSessionError:function:description:
  - OALTools, 214
- notifyContextDeallocating:
  - ALDevice, 49
- notifyContextInitializing:
  - ALDevice, 49
- notifyDeviceDeallocating:
  - OpenALManager, 221
- notifyDeviceInitializing:
  - OpenALManager, 221
- notifyExtAudioError:function:description:
  - OALTools, 214
- notifySourceDeallocating:
  - ALContext, 43
- notifySourceInitializing:
  - ALContext, 43
- notifyTrackDeallocating:
  - OALAudioTracks, 157
- notifyTrackInitializing:
  - OALAudioTracks, 157
- NSMutableArray, 113
- numberOfChannels
  - OALAudioTrack, 154
- numberOfLoops
  - OALAudioTrack, 154
- numObjects
  - OALCallAction, 161
- OAL\_AsyncALBufferLoadOperation, 114
  - initWithUrl:reduceToMono:target:selector:, 114
  - operationWithUrl:reduceToMono:target:selector:, 115
  - reduceToMono, 115
  - selector, 115
  - target, 115
  - url, 115
- OAL\_AsyncAudioTrackOperation, 116
  - audioTrack, 117
  - initWithTrack:url:seekTime:target:selector:, 117
  - operationWithTrack:url:seekTime:target:selector:, 117
  - seekTime, 117
  - selector, 117
  - target, 117
  - url, 117
- OAL\_AsyncAudioTrackPlayOperation, 118
  - initWithTrack:url:loops:target:selector:, 119
  - loops, 119
  - operationWithTrack:url:loops:target:selector:, 119
- OAL\_AsyncAudioTrackPreloadOperation, 120
- OAL\_GainProtocol-p, 120
  - gain, 120
- OAL\_PanProtocol-p, 121
  - pan, 121
- OAL\_PitchProtocol-p, 121
  - pitch, 121
- OAL\_PositionProtocol-p, 122
  - position, 122
- OALAction, 122
  - duration, 125
  - elapsed, 125
  - initWithDuration:, 124
  - prepareWithTarget:, 124
  - running, 125
  - runningInManager, 125
  - runWithTarget:, 124
  - startAction, 125
  - stopAction, 125
  - target, 125
  - updateCompletion:, 125
- OALActionManager, 126
  - actionsToAdd, 128
  - actionsToRemove, 128
  - doResetTimeDelta:, 127
  - lastTimestamp, 128
  - notifyActionStarted:, 127
  - notifyActionStopped:, 127
  - stepTimer, 128
  - stopAllActions, 127
  - SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, 127
  - targetActions, 128
  - targets, 128
- OALAudioFile, 128

- audioDataWithStartFrame:numFrames:bufferSize:closeOSResources, 145
- 130
- bufferFromUrl:reduceToMono:, 130
- bufferNamed:startFrame:numFrames:, 131
- close, 131
- closeOSResources, 131
- fileHandle, 132
- fileWithUrl:reduceToMono:, 131
- initWithUrl:reduceToMono:, 131
- originalChannelsPerFrame, 132
- reduceToMono, 132
- streamDescription, 132
- totalFrames, 132
- url, 132
- OALAudioSession, 133
  - allowIpod, 138
  - audioRoute, 138
  - audioSessionActive, 138
  - audioSessionCategory, 138
  - audioSessionDelegate, 139
  - audioSessionWasActive, 138
  - close, 135
  - closeOSResources, 135
  - forceEndInterruption, 136
  - getFloatProperty:, 136
  - getIntProperty:, 136
  - getStringProperty:, 136
  - handleInterruptions, 139
  - hardwareMuted, 139
  - hardwareVolume, 139
  - honorSilentSwitch, 139
  - ipodDucking, 139
  - ipodPlaying, 139
  - lastResetTime, 138
  - onAudioError:, 136
  - setAudioMode, 137
  - setIntProperty:value:, 137
  - setSuspended:, 137
  - suspendHandler, 138
  - SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, 137
  - updateFromAudioSessionCategory, 137
  - updateFromFlags, 137
  - useHardwareIfAvailable, 140
- OALAudioTrack, 140
  - autoPreload, 153
  - averagePowerForChannel:, 145
  - clear, 145
  - close, 145
  - currentlyLoadedUrl, 153
  - currentTime, 153
  - delegate, 153
  - deviceCurrentTime, 153
  - duration, 154
  - fadeTo:duration:target:selector:, 145
  - gain, 154
  - gainAction, 152
  - interrupted, 152
  - meteringEnabled, 154
  - muted, 154
  - numberOfChannels, 154
  - numberOfLoops, 154
  - operationQueue, 152
  - pan, 154
  - panAction, 152
  - panTo:duration:target:selector:, 146
  - paused, 154
  - peakPowerForChannel:, 146
  - play, 146
  - playAtTime:, 146
  - player, 154
  - playFile:, 146
  - playFile:loops:, 147
  - playFileAsync:loops:target:selector:, 147
  - playFileAsync:target:selector:, 147
  - playing, 155
  - playUrl:, 147
  - playUrl:loops:, 148
  - playUrlAsync:loops:target:selector:, 148
  - playUrlAsync:target:selector:, 148
  - preloaded, 155
  - preloadFile:, 148
  - preloadFile:seekTime:, 149
  - preloadFileAsync:seekTime:target:selector:, 149
  - preloadFileAsync:target:selector:, 149
  - preloadUrl:, 150
  - preloadUrl:seekTime:, 150
  - preloadUrlAsync:seekTime:target:selector:, 150
  - preloadUrlAsync:target:selector:, 151
  - setSuspended:, 151
  - simulatorPlayerRef, 152
  - stop, 151
  - stopActions, 151
  - stopFade, 151
  - stopPan, 151
  - suspendHandler, 153

- track, 152
- updateMeters, 152
- volume, 155
- OALAudioTracks, 155
  - close, 157
  - closeOSResources, 157
  - muted, 158
  - notifyTrackDeallocating:, 157
  - notifyTrackInitializing:, 157
  - paused, 158
  - suspendHandler, 157
- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, 157
- tracks, 157, 158
- OALCallAction, 158
  - actionWithCallTarget:selector:, 159
  - actionWithCallTarget:selector:withObject:, 160
  - actionWithCallTarget:selector:withObject:withObject:, 160
  - callTarget, 161
  - initWithCallTarget:selector:, 160
  - initWithCallTarget:selector:withObject:, 160
  - initWithCallTarget:selector:withObject:withObject:, 161
  - numObjects, 161
  - object1, 161
  - object2, 161
  - selector, 162
- OALConcurrentActions, 162
  - actions, 164
  - actions:, 163
  - actionsFromArray:, 163
  - actionsWithDuration, 164
  - initWithActions:, 164
  - pDurations, 164
- OALExponentialFunction, 164
  - function, 165
- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, 165
- OALFunction-p, 166
  - valueForInput:, 166
- OALFunctionAction, 167
  - actionWithDuration:endValue:, 168
  - actionWithDuration:endValue:function:, 169
  - actionWithDuration:startValue:endValue:function:, 169
  - defaultFunction, 169
  - delta, 171
  - endValue, 171
  - function, 171
  - initWithDuration:endValue:, 169
  - initWithDuration:endValue:function:, 170
  - initWithDuration:startValue:endValue:function:, 170
  - lowValue, 171
  - realFunction, 171
  - reverseFunction, 171
  - startValue, 171
- OALGainAction, 171
- OALLinearFunction, 172
  - function, 173
- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, 173
- OALLogarithmicFunction, 173
  - function, 174
- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, 174
- OALMoveByAction, 175
  - actionWithDuration:delta:, 176
  - actionWithUnitsPerSecond:delta:, 176
  - delta, 177
  - initWithDuration:delta:, 176
  - initWithUnitsPerSecond:delta:, 177
  - startPoint, 177
  - unitsPerSecond, 177
- OALMoveToAction, 178
  - actionWithDuration:position:, 179
  - actionWithUnitsPerSecond:position:, 179
  - delta, 180
  - initWithDuration:position:, 179
  - initWithUnitsPerSecond:position:, 180
  - position, 180
  - startPoint, 180
  - unitsPerSecond, 180
- OALPanAction, 181
- OALPitchAction, 181
- OALPlaceAction, 182
  - actionWithPosition:, 183
  - initWithPosition:, 183
  - position, 183
- OALReverseFunction, 183
  - function, 185
  - functionWithFunction:, 184
  - initWithFunction:, 185
- OALSCurveFunction, 185
  - function, 186

- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_preloadCacheCount, 202
- HEADER, 186
- OALSequentialActions, 187
  - actionIndex, 189
  - actions, 189
  - actions:, 188
  - actionsFromArray:, 188
  - currentAction, 189
  - initWithActions:, 188
  - pCurrentActionComplete, 189
  - pCurrentActionDuration, 189
  - pDurations, 189
  - pLastComplete, 189
- OALSimpleAudio, 190
  - allowIpod, 201
  - backgroundTrack, 201
  - backgroundTrackURL, 201
  - bgMuted, 201
  - bgPaused, 201
  - bgPlaying, 201
  - bgVolume, 201
  - channel, 200
  - close, 194
  - closeOSResources, 194
  - context, 200
  - device, 200
  - effectsMuted, 201
  - effectsPaused, 202
  - effectsVolume, 202
  - honorSilentSwitch, 202
  - initWithSources:, 194
  - internalPreloadEffect:reduceToMono:, 194
  - interrupted, 202
  - manuallySuspended, 202
  - muted, 202
  - paused, 202
  - pendingLoadCount, 200
  - playBg, 195
  - playBg:, 195
  - playBg:loop:, 195
  - playBg:volume:pan:loop:, 196
  - playBgWithLoop:, 196
  - playBuffer:volume:pitch:pan:loop:, 196
  - playEffect:, 197
  - playEffect:loop:, 197
  - playEffect:volume:pitch:pan:loop:, 197
  - preloadBg:, 198
  - preloadBg:seekTime:, 198
  - preloadCache, 200
  - preloadCacheCount, 202
  - preloadCacheEnabled, 202
  - preloadEffect:, 198
  - preloadEffect:reduceToMono:, 199
  - reservedSources, 203
  - resetToDefault, 199
  - sharedInstanceWithSources:, 199
  - stopAllEffects, 199
  - stopBg, 199
  - stopEverything, 199
  - suspended, 203
- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_ -
  - HEADER, 200
  - unloadAllEffects, 200
  - unloadEffect:, 200
  - useHardwareIfAvailable, 203
- OALSuspendHandler, 203
  - addSuspendListener:, 205
  - handlerWithTarget:selector:, 205
  - initWithTarget:selector:, 205
  - interrupted, 207
  - interruptLock, 206
  - listeners, 206
  - manuallySuspended, 207
  - manualSuspendLock, 206
  - manualSuspendStates, 206
  - removeSuspendListener:, 206
  - suspended, 207
  - suspendStatusChangeSelector, 206
  - suspendStatusChangeTarget, 207
- OALSuspendListener-p, 207
  - interrupted, 209
  - manuallySuspended, 209
- OALSuspendManager-p, 209
  - addSuspendListener:, 211
  - removeSuspendListener:, 211
  - suspended, 211
- OALTargetedAction, 211
  - action, 213
  - actionWithTarget:action:, 212
  - forcedTarget, 213
  - initWithTarget:action:, 212
- OALTools, 213
  - notifyAudioSessionError:function:description:, 214
  - notifyExtAudioError:function:description:, 214
  - urlForPath:, 214
- object1
  - OALCallAction, 161



- object2
  - OALCallAction, [161](#)
- offsetInBytes
  - ALSource, [73](#)
- offsetInSamples
  - ALSource, [73](#)
- offsetInSeconds
  - ALSource, [73](#)
- onAudioError:
  - OALAudioSession, [136](#)
- onFadeComplete:
  - ALChannelSource, [33](#)
- onPanComplete:
  - ALChannelSource, [33](#)
- onPitchComplete:
  - ALChannelSource, [33](#)
- OpenALManager, [215](#)
  - availableCaptureDevices, [222](#)
  - availableDevices, [222](#)
  - bufferAsyncFromFile:reduceToMono:target:selector:, [217](#)
  - bufferAsyncFromFile:target:selector:, [218](#)
  - bufferAsyncFromUrl:reduceToMono:target:selector:, [218](#)
  - bufferAsyncFromUrl:target:selector:, [219](#)
  - bufferFromFile:, [219](#)
  - bufferFromFile:reduceToMono:, [220](#)
  - bufferFromUrl:, [220](#)
  - bufferFromUrl:reduceToMono:, [220](#)
  - clearAllBuffers, [221](#)
  - close, [221](#)
  - closeOSResources, [221](#)
  - currentContext, [222](#)
  - defaultCaptureDeviceSpecifier, [222](#)
  - defaultDeviceSpecifier, [222](#)
  - devices, [221](#), [222](#)
  - mixerOutputFrequency, [222](#)
  - notifyDeviceDeallocating:, [221](#)
  - notifyDeviceInitializing:, [221](#)
  - operationQueue, [221](#)
  - setSuspended:, [221](#)
  - suspendHandler, [222](#)
  - SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_HEADER, [221](#)
- openCaptureDevice:frequency:format:bufferSize:
  - ALWrapper, [106](#)
- openDevice:
  - ALWrapper, [106](#)
- operationQueue
  - OALAudioTrack, [152](#)
- OpenALManager, [221](#)
- operationWithTrack:url:loops:target:selector:
  - OAL\_AsyncAudioTrackPlayOperation, [119](#)
- operationWithTrack:url:seekTime:target:selector:
  - OAL\_AsyncAudioTrackOperation, [117](#)
- operationWithUrl:reduceToMono:target:selector:
  - OAL\_AsyncALBufferLoadOperation, [115](#)
- orientation
  - ALListener, [53](#)
- originalChannelsPerFrame
  - OALAudioFile, [132](#)
- pan
  - ALSoundSource-p, [62](#)
  - OAL\_PanProtocol-p, [121](#)
  - OALAudioTrack, [154](#)
- panAction
  - ALSource, [72](#)
- panCompleteSelector
  - OALAudioTrack, [152](#)
- panCompleteTarget
  - ALChannelSource, [35](#)
- panTo:duration:target:selector:
  - ALSoundSource-p, [59](#)
  - OALAudioTrack, [146](#)
- paused
  - ALChannelSource, [36](#)
  - ALSoundSource-p, [62](#)
  - OALAudioTrack, [154](#)
  - OALAudioTracks, [158](#)
  - OALSimpleAudio, [202](#)
- pCurrentActionComplete
  - OALSequentialActions, [189](#)
- pCurrentActionDuration
  - OALSequentialActions, [189](#)
- pDurations
  - OALConcurrentActions, [164](#)
  - OALSequentialActions, [189](#)
- peakPowerForChannel:
  - OALAudioTrack, [146](#)
- pendingLoadCount
  - OALSimpleAudio, [200](#)
- pitchAction
  - ALSource, [72](#)

- pitchCompleteSelector
  - ALChannelSource, [36](#)
- pitchCompleteTarget
  - ALChannelSource, [36](#)
- pitchTo:duration:target:selector:
  - ALSoundSource-p, [59](#)
- pLastComplete
  - OALSequentialActions, [189](#)
- play
  - ALSource, [70](#)
  - OALAudioTrack, [146](#)
- play:
  - ALSoundSource-p, [59](#)
- play:gain:pitch:pan:loop:
  - ALSoundSource-p, [59](#)
- play:loop:
  - ALSoundSource-p, [60](#)
- playAtTime:
  - OALAudioTrack, [146](#)
- playBg
  - OALSimpleAudio, [195](#)
- playBg:
  - OALSimpleAudio, [195](#)
- playBg:loop:
  - OALSimpleAudio, [195](#)
- playBg:volume:pan:loop:
  - OALSimpleAudio, [196](#)
- playBgWithLoop:
  - OALSimpleAudio, [196](#)
- playBuffer:volume:pitch:pan:loop:
  - OALSimpleAudio, [196](#)
- playEffect:
  - OALSimpleAudio, [197](#)
- playEffect:loop:
  - OALSimpleAudio, [197](#)
- playEffect:volume:pitch:pan:loop:
  - OALSimpleAudio, [197](#)
- player
  - OALAudioTrack, [154](#)
- playFile:
  - OALAudioTrack, [146](#)
- playFile:loops:
  - OALAudioTrack, [147](#)
- playFileAsync:loops:target:selector:
  - OALAudioTrack, [147](#)
- playFileAsync:target:selector:
  - OALAudioTrack, [147](#)
- playing
  - ALSoundSource-p, [62](#)
  - OALAudioTrack, [155](#)
- playUrl:
  - OALAudioTrack, [147](#)
- playUrl:loops:
  - OALAudioTrack, [148](#)
- playUrlAsync:loops:target:selector:
  - OALAudioTrack, [148](#)
- playUrlAsync:target:selector:
  - OALAudioTrack, [148](#)
- pool
  - ALSoundSourcePool, [66](#)
- position
  - ALChannelSource, [36](#)
  - ALListener, [53](#)
  - ALSoundSource-p, [63](#)
  - OAL\_PositionProtocol-p, [122](#)
  - OALMoveToAction, [180](#)
  - OALPlaceAction, [183](#)
- preloadBg:
  - OALSimpleAudio, [198](#)
- preloadBg:seekTime:
  - OALSimpleAudio, [198](#)
- preloadCache
  - OALSimpleAudio, [200](#)
- preloadCacheCount
  - OALSimpleAudio, [202](#)
- preloadCacheEnabled
  - OALSimpleAudio, [202](#)
- preloaded
  - OALAudioTrack, [155](#)
- preloadEffect:
  - OALSimpleAudio, [198](#)
- preloadEffect:reduceToMono:
  - OALSimpleAudio, [199](#)
- preloadFile:
  - OALAudioTrack, [148](#)
- preloadFile:seekTime:
  - OALAudioTrack, [149](#)
- preloadFileAsync:seekTime:target:selector:
  - OALAudioTrack, [149](#)
- preloadFileAsync:target:selector:
  - OALAudioTrack, [149](#)
- preloadUrl:
  - OALAudioTrack, [150](#)
- preloadUrl:seekTime:
  - OALAudioTrack, [150](#)
- preloadUrlAsync:seekTime:target:selector:
  - OALAudioTrack, [150](#)
- preloadUrlAsync:target:selector:
  - OALAudioTrack, [151](#)
- prepareWithTarget:

- OALAction, [124](#)
- process
  - ALContext, [43](#)
- processContext:
  - ALWrapper, [106](#)
- queueBuffer:
  - ALSource, [70](#)
- queueBuffers:
  - ALSource, [70](#)
- realFunction
  - OALFunctionAction, [171](#)
- reduceToMono
  - OAL\_AsyncALBufferLoadOperation, [115](#)
  - OALAudioFile, [132](#)
- referenceDistance
  - ALChannelSource, [36](#)
  - ALSoundSource-p, [63](#)
- removeSource:
  - ALSoundSourcePool, [66](#)
- removeSuspendListener:
  - OALSuspendHandler, [206](#)
  - OALSuspendManager-p, [211](#)
- renderer
  - ALContext, [45](#)
- reservedSources
  - ALChannelSource, [37](#)
  - OALSimpleAudio, [203](#)
- resetToDefault
  - ALChannelSource, [33](#)
  - OALSimpleAudio, [199](#)
- reverseFunction
  - OALFunctionAction, [171](#)
- rewind
  - ALSoundSource-p, [60](#)
- rolloffFactor
  - ALChannelSource, [36](#)
  - ALSoundSource-p, [63](#)
- running
  - OALAction, [125](#)
- runningInManager
  - OALAction, [125](#)
- runWithTarget:
  - OALAction, [124](#)
- seekTime
  - OAL\_AsyncAudioTrackOperation, [117](#)
- selector
  - OAL\_AsyncALBufferLoadOperation, [115](#)
- OAL\_AsyncAudioTrackOperation, [117](#)
- OALCallAction, [162](#)
- setAudioMode
  - OALAudioSession, [137](#)
- setIntProperty:value:
  - OALAudioSession, [137](#)
- setMixerOutputDataRate:
  - ALWrapper, [106](#)
- setSuspended:
  - ALContext, [43](#)
  - ALSource, [70](#)
  - OALAudioSession, [137](#)
  - OALAudioTrack, [151](#)
  - OpenALManager, [221](#)
- shadowState
  - ALSource, [72](#)
- sharedInstanceWithSources:
  - OALSimpleAudio, [199](#)
- simulatorPlayerRef
  - OALAudioTrack, [152](#)
- size
  - ALBuffer, [24](#)
- source
  - ALSource, [70](#)
- source3f:parameter:v1:v2:v3:
  - ALWrapper, [107](#)
- source3i:parameter:v1:v2:v3:
  - ALWrapper, [107](#)
- sourcef:parameter:value:
  - ALWrapper, [107](#)
- sourcefv:parameter:values:
  - ALWrapper, [108](#)
- sourcei:parameter:value:
  - ALWrapper, [108](#)
- sourceid
  - ALSource, [73](#)
- sourceiv:parameter:values:
  - ALWrapper, [108](#)
- sourceOnContext:
  - ALSource, [70](#)
- sourcePause:
  - ALWrapper, [108](#)
- sourcePausev:numSources:
  - ALWrapper, [109](#)
- sourcePlay:
  - ALWrapper, [109](#)
- sourcePlayv:numSources:
  - ALWrapper, [109](#)
- sourcePool
  - ALChannelSource, [37](#)

- sourceQueueBuffers:numBuffers:bufferIds:
  - ALWrapper, 110
- sourceRelative
  - ALChannelSource, 36
  - ALSoundSource-p, 63
- sourceRewind:
  - ALWrapper, 110
- sourceRewindv:numSources:
  - ALWrapper, 110
- sources
  - ALContext, 44, 45
  - ALSoundSourcePool, 66
- sourceStop:
  - ALWrapper, 110
- sourceStopv:numSources:
  - ALWrapper, 111
- sourceType
  - ALChannelSource, 36
  - ALSoundSource-p, 63
- sourceUnqueueBuffers:numBuffers:bufferIds:
  - ALWrapper, 111
- speedOfSound
  - ALContext, 45
- speedOfSound:
  - ALWrapper, 111
- startAction
  - OALAction, 125
- startCapture
  - ALCaptureDevice, 28
- startCapture:
  - ALWrapper, 111
- startPoint
  - OALMoveByAction, 177
  - OALMoveToAction, 180
- startValue
  - OALFunctionAction, 171
- state
  - ALSource, 73
- stepTimer
  - OALActionManager, 128
- stop
  - ALSoundSource-p, 60
  - OALAudioTrack, 151
- stopAction
  - OALAction, 125
- stopActions
  - ALSoundSource-p, 60
  - OALAudioTrack, 151
- stopAllActions
  - OALActionManager, 127
- stopAllEffects
  - OALSimpleAudio, 199
- stopAllSounds
  - ALContext, 44
- stopBg
  - OALSimpleAudio, 199
- stopCapture
  - ALCaptureDevice, 28
- stopCapture:
  - ALWrapper, 112
- stopEverything
  - OALSimpleAudio, 199
- stopFade
  - ALSoundSource-p, 60
  - OALAudioTrack, 151
- stopPan
  - ALSoundSource-p, 60
  - OALAudioTrack, 151
- stopPitch
  - ALSoundSource-p, 61
- streamDescription
  - OALAudioFile, 132
- suspendContext:
  - ALWrapper, 112
- suspended
  - ALContext, 44
  - OALSimpleAudio, 203
  - OALSuspendHandler, 207
  - OALSuspendManager-p, 211
- suspendHandler
  - ALContext, 44
  - ALDevice, 49
  - ALListener, 52
  - ALSource, 72
  - OALAudioSession, 138
  - OALAudioTrack, 153
  - OALAudioTracks, 157
  - OpenALManager, 222
- suspendStatusChangeSelector
  - OALSuspendHandler, 206
- suspendStatusChangeTarget
  - OALSuspendHandler, 207
- SYNTHESIZE\_SINGLETON\_FOR\_CLASS\_ -
  - HEADER
  - IOSVersion, 113
  - OALActionManager, 127
  - OALAudioSession, 137
  - OALAudioTracks, 157
  - OALExponentialFunction, 165
  - OALLinearFunction, 173

- OALLogarithmicFunction, [174](#)
  - OALSCurveFunction, [186](#)
  - OALSimpleAudio, [200](#)
  - OpenALManager, [221](#)
- target
  - OAL\_AsyncALBufferLoadOperation, [115](#)
  - OAL\_AsyncAudioTrackOperation, [117](#)
  - OALAction, [125](#)
- targetActions
  - OALActionManager, [128](#)
- targets
  - OALActionManager, [128](#)
- totalFrames
  - OALAudioFile, [132](#)
- track
  - OALAudioTrack, [152](#)
- tracks
  - OALAudioTracks, [157](#), [158](#)
- unitsPerSecond
  - OALMoveByAction, [177](#)
  - OALMoveToAction, [180](#)
- unloadAllEffects
  - OALSimpleAudio, [200](#)
- unloadEffect:
  - OALSimpleAudio, [200](#)
- unqueueBuffer:
  - ALSource, [71](#)
- unqueueBuffers:
  - ALSource, [71](#)
- up
  - ALOrientation, [54](#)
- updateCompletion:
  - OALAction, [125](#)
- updateFromAudioSessionCategory
  - OALAudioSession, [137](#)
- updateFromFlags
  - OALAudioSession, [137](#)
- updateMeters
  - OALAudioTrack, [152](#)
- url
  - OAL\_AsyncALBufferLoadOperation, [115](#)
  - OAL\_AsyncAudioTrackOperation, [117](#)
  - OALAudioFile, [132](#)
- urlForPath:
  - OALTools, [214](#)
- useHardwareIfAvailable
  - OALAudioSession, [140](#)
  - OALSimpleAudio, [203](#)
- valueForInput:
  - OALFunction-p, [166](#)
- velocity
  - ALChannelSource, [37](#)
  - ALListener, [53](#)
  - ALSoundSource-p, [63](#)
- vendor
  - ALContext, [46](#)
- version
  - IOSVersion, [113](#)
- volume
  - ALSoundSource-p, [63](#)
  - OALAudioTrack, [155](#)
- x
  - ALPoint, [55](#)
  - ALVector, [74](#)
- y
  - ALPoint, [55](#)
  - ALVector, [74](#)
- z
  - ALPoint, [55](#)
  - ALVector, [74](#)