OpenAL Programmer's Guide

OpenAL Versions 1.0 and 1.1

Trademarks and Service Marks
Creative, Sound Blaster, Sound Blaster X-Fi, and the Creative logo are registered trademarks, and Environmental Audio, EAX, and the Environmental Audio Extensions logo are trademarks of Creative Technology Ltd. in the United States and/or other countries.

All other brands and product names listed are trademarks or registered trademarks of their respective holders.

Acknowledgments
Documentation written by Garin Hiebert. Additional input by Keith Charley, Peter Harrison, Jean-Marc Jot, Daniel Peacock, Jean-Michel Trivi, and Carlo Vogelsang.

Revision History

Revision 1.0	October 2005	Garin Hiebert
Revision 1.1	July 2006	Garin Hiebert
Revision 1.2	December 2006	Garin Hiebert
Revision 1.3	March 2007	Peter Harrison
Revision 1.4	June 2007	Peter Harrison
Revision 1.5	June 2007	Daniel Peacock

Table of Contents

About this Document

Introduction

OpenAL is a cross-platform three-dimensional audio API. The API's primary purpose is to allow an application to position audio sources in a three-dimensional space around a listener, producing reasonable spatialization of the sources for the audio system (headphones, 2.1 speaker output, 5.1 speaker output, etc.) Through extensions, Creative Labs has also enhanced OpenAL with EAX and other capabilities. OpenAL is appropriate for many audio applications, but was designed to be most appropriate for gaming audio.

Intended Audience

This reference guide is most appropriate for a programmer. Experience with C or C++ is not required to learn the concepts in OpenAL, but will make understanding the OpenAL source as well as sample code easier. Since there are several sample applications included with the OpenAL SDKs as well as with the source distribution, it is recommended that interested programmers take advantage of those resources.

Other OpenAL Resources

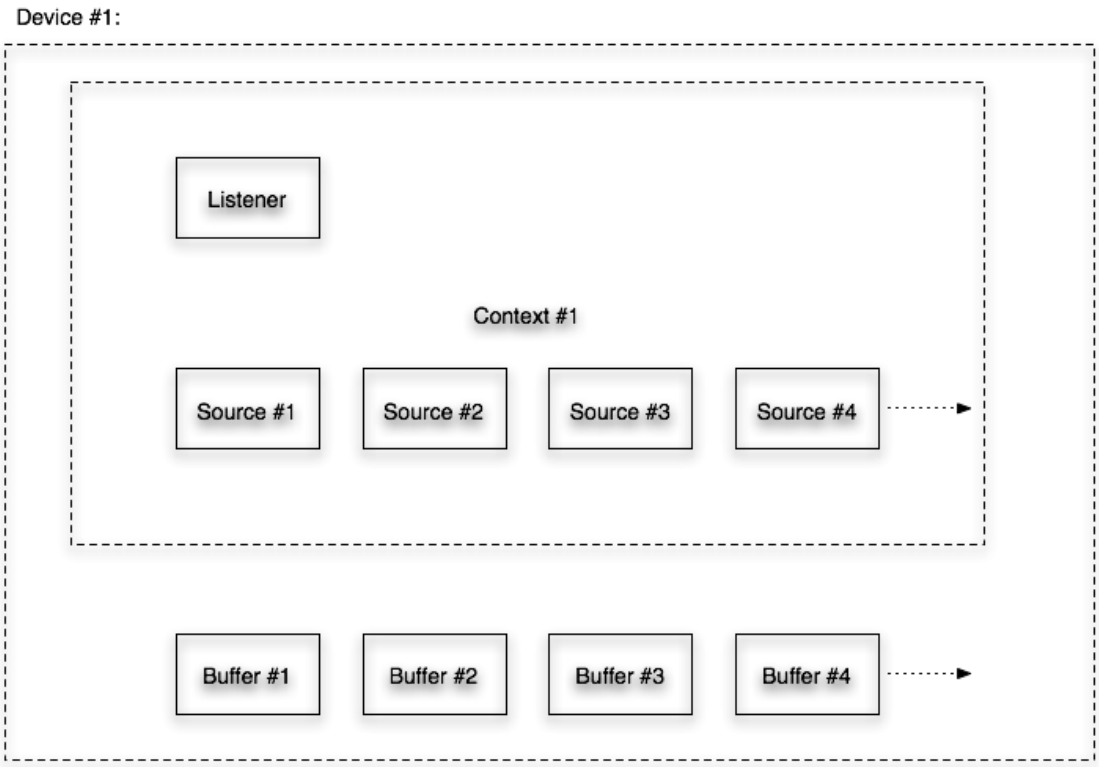
The two most important resources for additional information on OpenAL are the websites at *www.openal.org* and *http://developer.creative.com*. The main OpenAL site hosts the specification, the open source implementations, and sample code. The Creative developer site has a section dedicated to OpenAL with SDKs showing how to use OpenAL as well as various extensions.

Introduction to OpenAL

Use of OpenAL revolves around the use of three fundamental objects – Buffers, Sources, and a Listener. A buffer can be filled with audio data, and can then be attached to a source. The source can then be positioned and played. How the source is heard is determined by its position and orientation relative to the Listener object (there is only one Listener). Creating a number of sources and buffers and a single listener and then updating the positions and orientations of the sources and listener dynamically can present a convincing 3D audio world.

Objects

Here is a diagram showing the fundamental OpenAL objects and their relationships to the context and device objects:



When initializing OpenAL, at least one device has to be opened. Within that device, at least one context will be created. Within that context, one listener object is implied, and a multitude of source objects can be created. Each source can have one or more buffers objects attached to it. Buffer objects are not part of a specific context – they are shared among all contexts on one device.

Device Enumeration

The function call to open a device, alcOpenDevice, takes a string as input. The string should contain either the name of a valid OpenAL rendering device, or NULL to request the default device.

On PC Systems, a number of different OpenAL renderering devices may co-exist. For example a "native" renderer specific to the user's high-end soundcard, and a host-based software fallback renderer. On platforms where multiple renderers can be present, an OpenAL application may require the ability to identify the different devices available, in order to give the end-user a choice of device. OpenAL's Enumeration extension makes this possible.

The Enumeration extension allows the programmer to retrieve a string listing the names of available devices. It can also provide the name of the default device. Use alcGetString with the device property set to NULL, and the enum property set to ALC_DEVICE_SPECIFIER to get the list of available devices. To get the default device name, pass in NULL and ALC_DEFAULT_DEVICE_SPECIFIER .

The Enumeration extension also works with capture devices – the equivalent values are ALC_CAPTURE_DEVICE_SPECIFIER and ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER.

The programmer can find out more about the capabilities of each device by querying to see which extensions it supports using alcIsExtensionPresent and alIsExtensionPresent.

Initializing/Exiting

As described above, the first step to initializing OpenAL is to open a device. Once that is successfully done, then a context is opened on that device. Now the fundamental OpenAL objects can be managed – the listener, various sources, and various buffers.

To generate a set of buffers for use, use alGetError to reset the error state, call alGenBuffers to generate the number of buffers desired, and then use alGetError again to detect if an error was generated.

Fill the buffers with PCM data using alBufferData.

To generate a set of sources for use, use alGetError to reset the error state, call alGenSources to generate the number of sources desired, and then use alGetError again to detect if an error was generated.

Buffers are attached to sources using alSourcei.

Once a buffer has been attached to a source, the source can play the buffer using alSourcePlay.

Source and Listener properties can be updated dynamically using property set and get calls such as alGetListenerfv, alListener3f, alSourcei, and alGetSource3f.

Example:

```
// Initialization
Device = alcOpenDevice(NULL); // select the "preferred device"

if (Device) {
      Context=alcCreateContext(Device,NULL);
      alcMakeContextCurrent(Context);
}

// Check for EAX 2.0 support
g_bEAX = alIsExtensionPresent("EAX2.0");

// Generate Buffers
alGetError(); // clear error code

alGenBuffers(NUM_BUFFERS, g_Buffers);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alGenBuffers :", error);
      return;
}

// Load test.wav
loadWAVFile("test.wav",&format,&data,&size,&freq,&loop);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alutLoadWAVFile test.wav : ", error);
      alDeleteBuffers(NUM_BUFFERS, g_Buffers);
      return;
}

// Copy test.wav data into AL Buffer 0
alBufferData(g_Buffers[0],format,data,size,freq);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alBufferData buffer 0 : ", error);
      alDeleteBuffers(NUM_BUFFERS, g_Buffers);
      return;
}

// Unload test.wav
unloadWAV(format,data,size,freq);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alutUnloadWAV : ", error);
      alDeleteBuffers(NUM_BUFFERS, g_Buffers);
      return;
}

// Generate Sources
alGenSources(1,source);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alGenSources 1 : ", error);
      return;
}

// Attach buffer 0 to source
```

```
alSourcei(source[0], AL_BUFFER, g_Buffers[0]);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alSourcei AL_BUFFER 0 : ", error);
}

// Exit
Context=alcGetCurrentContext();
Device=alcGetContextsDevice(Context);
alcMakeContextCurrent(NULL);
alcDestroyContext(Context);
alcCloseDevice(Device);
```

Listener Properties

For every context, there is automatically one Listener object. The *alListener[f, 3f, fv, i]* and *alGetListener[f, 3f, fv, i]* families of functions can be used to set or retrieve the following listener properties:

Property	Data Type	Description
AL_GAIN	f, fv	“master gain” value should be positive
AL_POSITION	fv, 3f, iv, 3i	X, Y, Z position
AL_VELOCITY	fv, 3f, iv, 3i	velocity vector
AL_ORIENTATION	fv, iv	orientation expressed as “at” and “up” vectors

Example:

```
ALfloat listenerPos[]={0.0,0.0,0.0};
ALfloat listenerVel[]={0.0,0.0,0.0};
ALfloat listenerOri[]={0.0,0.0,-1.0, 0.0,1.0,0.0};

// Position ...
alListenerfv(AL_POSITION,listenerPos);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alListenerfv POSITION : ", error);
      return;
}

// Velocity ...
alListenerfv(AL_VELOCITY,listenerVel);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alListenerfv VELOCITY : ", error);
      return;
}

// Orientation ...
alListenerfv(AL_ORIENTATION,listenerOri);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alListenerfv ORIENTATION : ", error);
      return;
}
```

Buffer Properties

Each buffer generated by alGenBuffers has properties which can be retrieved. The *alGetBuffer[f, i]* functions can be used to retrieve the following buffer properties:

Property	Data Type	Description
AL_ FREQUENCY	i, iv	frequency of buffer in Hz
AL_ BITS	i, iv	bit depth of buffer
AL_ CHANNELS	i, iv	number of channels in buffer > 1 is valid, but buffer won't be positioned when played
AL_ SIZE	i, iv	size of buffer in bytes
AL_DATA	i, iv	original location where data was copied from generally useless, as was probably freed after buffer creation

Example:

```
// Retrieve Buffer Frequency
alBufferi(g_Buffers[0], AL_FREQUENCY, iFreq);
```

Source Properties

Each source generated by alGenSources has properties which can be set or retrieved. The *alSource[f, 3f, fv, i]* and *alGetSource[f, 3f, fv, i]* families of functions can be used to set or retrieve the following source properties:

Property	Data Type	Description
AL_PITCH	f, fv	pitch multiplier always positive
AL_GAIN	f, fv	source gain value should be positive
AL_MAX_DISTANCE	f, fv, i, iv	used with the Inverse Clamped Distance Model to set the distance where there will no longer be any attenuation of the source
AL_ROLLOFF_FACTOR	f, fv, i, iv	the rolloff rate for the source default is 1.0
AL_REFERENCE_DISTANCE	f, fv, i, iv	the distance under which the volume for the source would normally drop by half (before being influenced by rolloff factor or AL_MAX_DISTANCE)
AL_MIN_GAIN	f, fv	the minimum gain for this source
AL_MAX_GAIN	f, fv	the maximum gain for this source
AL_CONE_OUTER_GAIN	f, fv	the gain when outside the oriented cone
AL_CONE_INNER_ANGLE	f, fv, i, iv	the gain when inside the oriented cone
AL_CONE_OUTER_ANGLE	f, fv, i, iv	outer angle of the sound cone, in degrees default is 360
AL_POSITION	fv, 3f	X, Y, Z position
AL_VELOCITY	fv, 3f	velocity vector
AL_DIRECTION	fv, 3f, iv, 3i	direction vector
AL_SOURCE_RELATIVE	i, iv	determines if the positions are relative to the listener default is AL_FALSE
AL_SOURCE_TYPE	i, iv	the soruce type – AL_UNDETERMINED, AL_STATIC, or AL_STREAMING
AL_LOOPING	i, iv	turns looping on (AL_TRUE) or off (AL_FALSE)
AL_BUFFER	i, iv	the ID of the attached buffer
AL_SOURCE_STATE	i, iv	the state of the source (AL_STOPPED, AL_PLAYING, …)
AL_BUFFERS_QUEUED*	i, iv	the number of buffers queued on this source
AL_BUFFERS_PROCESSED*	i, iv	the number of buffers in the queue that have been processed
AL_SEC_OFFSET	f, fv, i, iv	the playback position, expressed in seconds
AL_SAMPLE_OFFSET	f, fv, i, iv	the playback position, expressed in samples
AL_BYTE_OFFSET	f, fv, i, iv	the playback position, expressed in bytes

* *Read Only (alGetSourcei)*

Example:

```
alGetError(); // clear error state
alSourcef(source[0],AL_PITCH,1.0f);
if ((error = alGetError()) != AL_NO_ERROR)
      DisplayALError("alSourcef 0 AL_PITCH : \n", error);

alGetError(); // clear error state
alSourcef(source[0],AL_GAIN,1.0f);
if ((error = alGetError()) != AL_NO_ERROR)
      DisplayALError("alSourcef 0 AL_GAIN : \n", error);

alGetError(); // clear error state
alSourcefv(source[0],AL_POSITION,source0Pos);
if ((error = alGetError()) != AL_NO_ERROR)
      DisplayALError("alSourcefv 0 AL_POSITION : \n", error);
```

```
alGetError(); // clear error state
alSourcefv(source[0],AL_VELOCITY,source0Vel);
if ((error = alGetError()) != AL_NO_ERROR)
      DisplayALError("alSourcefv 0 AL_VELOCITY : \n", error);

alGetError(); // clear error state
alSourcei(source[0],AL_LOOPING,AL_FALSE);
if ((error = alGetError()) != AL_NO_ERROR)
      DisplayALError("alSourcei 0 AL_LOOPING true: \n", error);
```

Queuing Buffers on a Source

To continuously stream audio from a source without interruption, buffer queuing is required. To use buffer queuing, the buffers and sources are generated in the normal way, but alSourcei is not used to attach the buffers to the source. Instead, the functions alSourceQueueBuffers and alSourceUnqueueBuffers are used. The program can attach a buffer or a set of buffers to a source using alSourceQueueBuffers, and then call alSourcePlay on that source. While the source is playing, alSourceUnqueueBuffers can be called to remove buffers which have already played. Those buffers can then be filled with new data or discarded. New or refilled buffers can then be attached to the playing source using alSourceQueueBuffers. As long as there is always a new buffer to play in the queue, the source will continue to play.

Although some 1.0 implementations of OpenAL may not enforce the following restrictions on queuing, it is recommended to observe the following additional rules, which do universally apply to 1.1 implementations:

1) A source that will be used for streaming should not have its first buffer attached using alSourcei – always use alSourceQueueBuffers to attach buffers to streaming sources. Any source can have all buffers detached from it using alSourcei(..., AL_BUFFER, 0), and can then be used for either streaming or non-streaming buffers depending on how data is then attached to the source (with alSourcei or with alSourceQueueBuffers).

2) All buffers attached to a source using alSourceQueueBuffers should have the same audio format.

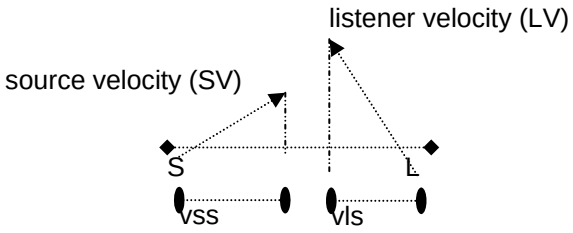
Doppler Shift

The Doppler effect depends on the velocities of source and listener relative to the medium, and the propagation speed of sound in that medium. The application might want to emphasize or de-emphasize the Doppler effect as physically accurate calculation might not give the desired results. The amount of frequency shift (pitch change) is proportional to the speed of listener and source along their line of sight.

The Doppler effect as implemented by OpenAL is described by the formula below. Effects of the medium (air, water) moving with respect to listener and source are ignored.

SS: AL_SPEED_OF_SOUND = speed of sound (default value 343.3)
DF: AL_DOPPLER_FACTOR = Doppler factor (default 1.0)
vls: Listener velocity scalar (scalar, projected on source-to-listener vector)
vss: Source velocity scalar (scalar, projected on source-to-listener vector)
f: Frequency of sample
f': effective Doppler shifted frequency

Graphic representation of vls and vss:



3D Mathematical representation of vls and vss:

Mag(vector) = sqrt(vector.x * vector.x + vector.y * vector.y + vector.z * vector.z)
DotProduct(v1, v2) = (v1.x * v2.x + v1.y * v2.y + v1.z * v2.z)
SL = source to listener vector
SV = Source Velocity vector
LV = Listener Velocity vector

vls = DotProduct(SL, LV) / Mag(SL)
vss = DotProduct(SL, SV) / Mag(SL)

Dopper Calculation:

vss = min(vss, SS/DF)
vls = min(vls, SS/DF)
f' = f * (SS - DF*vls) / (SS - DF*vss)

There are two API calls global to the current context that provide control of the speed of sound and Doppler factor. AL_DOPPLER_FACTOR is a simple scaling of source and listener velocities to exaggerate or deemphasize the Doppler (pitch) shift resulting from the calculation.

void alDopplerFactor(ALfloat dopplerFactor);

A negative value will result in an AL_INVALID_VALUE error, the command is then ignored. The default value is 1. The current setting can be queried using *alGetFloat{v}* and AL_DOPPLER_FACTOR.

AL_SPEED_OF_SOUND allows the application to change the reference (propagation) speed used in the Doppler calculation. The source and listener velocities should be expressed in the same units as the speed of sound.

void alSpeedOfSound(ALfloat speed);

A negative or zero value will result in an AL_INVALID_VALUE error, and the command is ignored. The default value is 343.3 (appropriate for velocity units of meters and air as the propagation medium). The current setting can be queried using *alGetFloat{v}* and AL_SPEED_OF_SOUND.

Distance and velocity units are completely independent of one another (so you could use different units for each if desired). If an OpenAL application doesn't want to use Doppler effects, then leaving all velocities at zero will achieve that result.

Error Handling

The error state of OpenAL can be retrieved at any time using alGetError. alGetError clears the error state of OpenAL when it is called, so it is common for an OpenAL application to call alGetError at the beginning of a critical operation to clear the error state, perform the critical operation, and then use alGetError again to test whether or not an error occurred.

Error Codes:

Error Code	Description
AL_NO_ERROR	there is not currently an error
AL_INVALID_NAME	a bad name (ID) was passed to an OpenAL function
AL_INVALID_ENUM	an invalid enum value was passed to an OpenAL function
AL_INVALID_VALUE	an invalid value was passed to an OpenAL function
AL_INVALID_OPERATION	the requested operation is not valid
AL_OUT_OF_MEMORY	the requested operation resulted in OpenAL running out of memory

Example:

```
alGetError();  // Clear Error Code

// Generate Buffers
alGenBuffers(NUM_BUFFERS, g_Buffers);
if ((error = alGetError()) != AL_NO_ERROR)
{
      DisplayALError("alGenBuffers :", error);
      exit(-1);
}
```

Extensions

OpenAL has an extension mechanism that can be used by OpenAL vendors to add new features to the API. Creative Labs have added a number of extensions including EAX, X-RAM, Multi-Channel Buffer playback, and most recently an Effect Extension (EFX). To determine if an extension is available the application can use either alIsExtensionPresent or alcIsExtensionPresent depending on the type of extension. The Appendices contain more details about some of Creative's extensions to OpenAL.

Buffer Functions

Properties

Property	Data Type	Description
AL_ FREQUENCY	i, iv	frequency of buffer in Hz
AL_ BITS	i, iv	bit depth of buffer
AL_ CHANNELS	i, iv	number of channels in buffer > 1 is valid, but buffer won't be positioned when played
AL_ SIZE	i, iv	size of buffer in bytes
AL_DATA	i, iv	original location where data was copied from generally useless, as was probably freed after buffer creation

Functions

alGenBuffers
alDeleteBuffers
alIsBuffer
alBufferData
alBufferf
alBuffer3f
alBufferfv
alBufferi
alBuffer3i
alBufferiv
alGetBufferf
alGetBuffer3f

alGenBuffers

Description

This function generates one or more buffers, which contain audio data (see alBufferData). References to buffers are ALuint values, which are used wherever a buffer reference is needed (in calls such as alDeleteBuffers, alSourcei, alSourceQueueBuffers, and alSourceUnqueueBuffers).

```
void alGenBuffers(
      ALsizei n,
      ALuint *buffers
);
```

Parameters

n	the number of buffers to be generated
buffers	pointer to an array of ALuint values which will store the names of the new buffers

Possible Error States

State	*Description*
AL_INVALID_VALUE	The buffer array isn't large enough to hold the number of buffers requested.
AL_OUT_OF_MEMORY	There is not enough memory available to generate all the buffers requested.

Version Requirements

OpenAL 1.0 or higher

Remarks

If the requested number of buffers cannot be created, an error will be generated which can be detected with alGetError. If an error occurs, no buffers will be generated. If n equals zero, alGenBuffers does nothing and does not return an error.

See Also

alDeleteBuffers, alIsBuffer

alDeleteBuffers

Description

This function deletes one or more buffers, freeing the resources used by the buffer. Buffers which are attached to a source can not be deleted. See alSourcei and alSourceUnqueueBuffers for information on how to detach a buffer from a source.

```
void alDeleteBuffers(
      ALsizei n,
      ALuint *buffers
);
```

Parameters

n the number of buffers to be deleted

buffers pointer to an array of buffer names identifying the buffers to be deleted

Possible Error States

State	*Description*
AL_INVALID_OPERATION	The buffer is still in use and can not be deleted.
AL_INVALID_NAME	A buffer name is invalid.
AL_INVALID_VALUE	The requested number of buffers can not be deleted.

Version Requirements

OpenAL 1.0 or higher

Remarks

If the requested number of buffers cannot be deleted, an error will be generated which can be detected with alGetError. If an error occurs, no buffers will be deleted. If n equals zero, alDeleteBuffers does nothing and will not return an error.

See Also

alGenBuffers, alIsBuffer

alIsBuffer

Description

This function tests if a buffer name is valid, returning AL_TRUE if valid, AL_FALSE if not.

```
ALboolean alIsBuffer(
      ALuint buffer
);
```

Parameters

buffer	a buffer name to be tested for validity

Possible Error States

None

Version Requirements

OpenAL 1.0 or higher

Remarks

The NULL buffer is always valid (see alSourcei for information on how the NULL buffer is used).

See Also

alGenBuffers, alDeleteBuffers

alBufferData

Description

This function fills a buffer with audio data. All the pre-defined formats are PCM data, but this function may be used by extensions to load other data types as well.

```
void alBufferData(
      ALuint buffer,
      ALenum format,
      const ALvoid *data,
      ALsizei size,
      ALsizei freq
);
```

Parameters

buffer	buffer name to be filled with data
format	format type from among the following: AL_FORMAT_MONO8 AL_FORMAT_MONO16 AL_FORMAT_STEREO8 AL_FORMAT_STEREO16
data	pointer to the audio data
size	the size of the audio data in bytes
freq	the frequency of the audio data

Possible Error States

State	***Description***
AL_OUT_OF_MEMORY	There is not enough memory available to create this buffer.
AL_INVALID_VALUE	The size parameter is not valid for the format specified, the buffer is in use, or the data is a NULL pointer.
AL_INVALID_ENUM	The specified format does not exist.

Version Requirements

OpenAL 1.0 or higher

Remarks

8-bit PCM data is expressed as an unsigned value over the range 0 to 255, 128 being an audio output level of zero. 16-bit PCM data is expressed as a signed value over the range -32768 to 32767, 0 being an audio output level of zero. Stereo data is expressed in interleaved format, left channel first. Buffers containing more than one channel of data will be played without 3D spatialization.

alBufferf

Description

This function sets a floating point property of a buffer.

```
void alBufferf(
      ALuint buffer,
      ALenum param,
      ALfloat value
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
param	the name of the attribute to be set
value	the ALfloat value to be set

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be affected by this call, but this function may be used by OpenAL extensions.

See Also

alBuffer3f, alBufferfv, alGetBufferf, alGetBuffer3f, alGetBufferfv

alBuffer3f

Description

This function sets a floating point property of a buffer.

```
void alBuffer3f(
      ALuint buffer,
      ALenum param,
      ALfloat v1,
      ALfloat v2,
      ALfloat v3
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
param	the name of the attribute to be set
v1, v2, v3	the ALfloat values to be set

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be affected by this call, but this function may be used by OpenAL extensions.

See Also

alBufferf, alBufferfv, alGetBufferf, alGetBuffer3f, alGetBufferfv

alBufferfv

Description

This function sets a floating point property of a buffer.

```
void alBufferfv(
      ALuint buffer,
      ALenum param,
      ALfloat *values
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
param	the name of the attribute to be set
values	a pointer to the ALfloat values to be set

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be affected by this call, but this function may be used by OpenAL extensions.

See Also

alBufferf, alBuffer3f, alGetBufferf, alGetBuffer3f, alGetBufferfv

alBufferi

Description

This function retrieves an integer property of a buffer.

```
void alBufferi(
      ALuint buffer,
      ALenum param,
      ALint value
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
param	the name of the attribute to be set
value	a pointer to an ALint to hold the retrieved data

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be affected by this call, but this function may be used by OpenAL extensions.

See Also

alBuffer3i, alBufferiv, alGetBufferi, alGetBuffer3i, alGetBufferiv

alBuffer3i

Description

This function sets a floating point property of a buffer.

```
void alBuffer3i(
      ALuint buffer,
      ALenum param,
      ALint v1,
      ALint v2,
      ALint v3
);
```

Parameters

buffer buffer name whose attribute is being retrieved

param the name of the attribute to be set

v1, v2, v3 the ALint values to be set

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be affected by this call, but this function may be used by OpenAL extensions.

See Also

alBufferi, alBufferiv, alGetBufferi, alGetBuffer3i, alGetBufferiv

alBufferiv

Description

This function sets a floating point property of a buffer.

```
void alBufferiv(
      ALuint buffer,
      ALenum param,
      ALint *values
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
param	the name of the attribute to be set
values	a pointer to the ALint values to be set

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be affected by this call, but this function may be used by OpenAL extensions.

See Also

alBufferi, alBuffer3i, alGetBufferi, alGetBuffer3i, alGetBufferiv

alGetBufferf

Description

This function retrieves a floating point property of a buffer.

```
void alGetBufferf(
      ALuint buffer,
      ALenum pname,
      ALfloat *value
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
pname	the name of the attribute to be retrieved
value	a pointer to an ALfloat to hold the retrieved data

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.
AL_INVALID_VALUE	The specified value pointer is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be retrieved by this call, but this function may be used by OpenAL extensions.

See Also

alBufferf, alBuffer3f, alBufferfv, alGetBuffer3f, alGetBufferfv

alGetBuffer3f

Description

This function retrieves a floating point property of a buffer.

```
void alGetBuffer3f(
      ALuint buffer,
      ALenum pname,
      ALfloat *v1,
      ALfloat *v2,
      ALfloat *v3
);
```

Parameters

buffer buffer name whose attribute is being retrieved

pname the name of the attribute to be retrieved

v1, v2, v3 pointers to a ALfloat values to hold the retrieved data

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.
AL_INVALID_VALUE	The specified value pointer is not valid.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be retrieved by this call, but this function may be used by OpenAL extensions.

See Also

alBufferf, alBuffer3f, alBufferfv, alGetBufferf, alGetBufferfv

alGetBufferfv

Description

This function retrieves a floating point property of a buffer.

```
void alGetBufferfv(
      ALuint buffer,
      ALenum pname,
      ALfloat *values
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
pname	the name of the attribute to be retrieved
values	pointer to an ALfloat vector to hold the retrieved data

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.
AL_INVALID_VALUE	The specified value pointer is not valid.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be retrieved by this call, but this function may be used by OpenAL extensions.

See Also

alBufferf, alBuffer3f, alBufferfv, alGetBufferf, alGetBuffer3f

alGetBufferi

Description

This function retrieves an integer property of a buffer.

```
void alGetBufferi(
      ALuint buffer,
      ALenum pname,
      ALint *value
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
pname	the name of the attribute to be retrieved: AL_FREQUENCY AL_BITS AL_CHANNELS AL_SIZE AL_DATA
value	a pointer to an ALint to hold the retrieved data

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.
AL_INVALID_VALUE	The specified value pointer is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alBufferi, alBuffer3i, alBufferiv, alGetBuffer3i, alGetBufferiv

alGetBuffer3i

Description

This function retrieves a floating point property of a buffer.

```
void alGetBuffer3i(
      ALuint buffer,
      ALenum pname,
      ALint *v1,
      ALint *v2,
      ALint *v3
);
```

Parameters

buffer buffer name whose attribute is being retrieved

pname the name of the attribute to be retrieved

v1, v2, v3 pointers to ALint values to hold the retrieved data

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.
AL_INVALID_VALUE	The specified value pointer is not valid.

Version Requirements

OpenAL 1.1 or higher

Remarks

There are no relevant buffer properties defined in OpenAL 1.1 which can be retrieved by this call, but this function may be used by OpenAL extensions.

See Also

alBufferi, alBuffer3i, alBufferiv, alGetBufferi, alGetBufferiv

alGetBufferiv

Description

This function retrieves a floating point property of a buffer.

```
void alGetBufferiv(
      ALuint buffer,
      ALenum pname,
      ALint *values
);
```

Parameters

buffer	buffer name whose attribute is being retrieved
pname	the name of the attribute to be retrieved: AL_FREQUENCY AL_BITS AL_CHANNELS AL_SIZE AL_DATA
values	pointer to an ALint vector to hold the retrieved data

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified buffer doesn't have parameters (the NULL buffer), or doesn't exist.
AL_INVALID_VALUE	The specified value pointer is not valid.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alBufferi, alBuffer3i, alBufferiv, alGetBufferi, alGetBuffer3i

Source Functions

Properties

Property	Data Type	Description
AL_PITCH	f, fv	pitch multiplier always positive
AL_GAIN	f, fv	source gain value should be positive
AL_MAX_DISTANCE	f, fv, i, iv	used with the Inverse Clamped Distance Model to set the distance where there will no longer be any attenuation of the source
AL_ROLLOFF_FACTOR	f, fv, i, iv	the rolloff rate for the source default is 1.0
AL_REFERENCE_DISTANCE	f, fv, i, iv	the distance under which the volume for the source would normally drop by half (before being influenced by rolloff factor or AL_MAX_DISTANCE)
AL_MIN_GAIN	f, fv	the minimum gain for this source
AL_MAX_GAIN	f, fv	the maximum gain for this source
AL_CONE_OUTER_GAIN	f, fv	the gain when outside the oriented cone
AL_CONE_INNER_ANGLE	f, fv, i, iv	the gain when inside the oriented cone
AL_CONE_OUTER_ANGLE	f, fv, i, iv	outer angle of the sound cone, in degrees default is 360
AL_POSITION	fv, 3f	X, Y, Z position
AL_VELOCITY	fv, 3f	velocity vector
AL_DIRECTION	fv, 3f, iv, 3i	direction vector
AL_SOURCE_RELATIVE	i, iv	determines if the positions are relative to the listener default is AL_FALSE
AL_SOURCE_TYPE	i, iv	the soruce type – AL_UNDETERMINED, AL_STATIC, or AL_STREAMING
AL_LOOPING	i, iv	turns looping on (AL_TRUE) or off (AL_FALSE)
AL_BUFFER	i, iv	the ID of the attached buffer
AL_SOURCE_STATE	i, iv	the state of the source (AL_STOPPED, AL_PLAYING, …)
AL_BUFFERS_QUEUED*	i, iv	the number of buffers queued on this source
AL_BUFFERS_PROCESSED*	i, iv	the number of buffers in the queue that have been processed
AL_SEC_OFFSET	f, fv, i, iv	the playback position, expressed in seconds
AL_SAMPLE_OFFSET	f, fv, i, iv	the playback position, expressed in samples
AL_BYTE_OFFSET	f, fv, i, iv	the playback position, expressed in bytes

Functions

alGenSources
alDeleteSources
alIsSource
alSourcef
alSource3f
alSourcefv
alSourcei
alSource3i

alSourceiv
alGetSourcef
alGetSource3f
alGetSourcefv
alGetSourcei
alGetSource3i
alGetSourceiv
alSourcePlay
alSourcePlayv
alSourcePause
alSourcePausev
alSourceStop
alSourceStopv
alSourceRewind
alSourceRewindv
alSourceQueueBuffers
alSourceUnqueueBuffers

alGenSources

Description

This function generates one or more sources. References to sources are ALuint values, which are used wherever a source reference is needed (in calls such as alDeleteSources and alSourcei).

```
void alGenSources(
      ALsizei n,
      ALuint *sources
);
```

Parameters

n	the number of sources to be generated
sources	pointer to an array of ALuint values which will store the names of the new sources

Possible Error States

State	*Description*
AL_OUT_OF_MEMORY	There is not enough memory to generate all the requested sources.
AL_INVALID_VALUE	There are not enough non-memory resources to create all the requested sources, or the array pointer is not valid.
AL_INVALID_OPERATION	There is no context to create sources in.

Version Requirements

OpenAL 1.0 or higher

Remarks

If the requested number of sources cannot be created, an error will be generated which can be detected with alGetError. If an error occurs, no sources will be generated. If n equals zero, alGenSources does nothing and does not return an error.

See Also

alDeleteSources, alIsSource

alDeleteSources

Description

This function deletes one or more sources.

```
void alDeleteSources(
      ALsizei n,
      ALuint *sources
);
```

Parameters

n — the number of sources to be deleted

sources — pointer to an array of source names identifying the sources to be deleted

Possible Error States

State	*Description*
AL_INVALID_NAME	At least one specified source is not valid, or an attempt is being made to delete more sources than exist.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

If the requested number of sources cannot be deleted, an error will be generated which can be detected with alGetError. If an error occurs, no sources will be deleted. If n equals zero, alDeleteSources does nothing and will not return an error.

A playing source can be deleted – the source will be stopped and then deleted.

See Also

alGenSources, alIsSource

alIsSource

Description

This function tests if a source name is valid, returning AL_TRUE if valid and AL_FALSE if not.

```
boolean alIsSource(
      ALuint source
);
```

Parameters

source a source name to be tested for validity

Possible Error States

State	*Description*
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alGenSources, alDeleteSources

alSourcef

Description

This function sets a floating point property of a source.

```
void alSourcef(
      ALuint source,
      ALenum param,
      ALfloat value
);
```

Parameters

source source name whose attribute is being set

param the name of the attribute to set:
AL_PITCH
AL_GAIN
AL_MIN_GAIN
AL_MAX_GAIN
AL_MAX_DISTANCE
AL_ROLLOFF_FACTOR
AL_CONE_OUTER_GAIN
AL_CONE_INNER_ANGLE
AL_CONE_OUTER_ANGLE
AL_REFERENCE_DISTANCE

value the value to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is out of range.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alSource3f, alSourcefv, alGetSourcef, alGetSource3f, alGetSourcefv

alSource3f

Description

This function sets a source property requiring three floating point values.

```
void alSource3f(
      ALuint source,
      ALenum param,
      ALfloat v1,
      ALfloat v2,
      ALfloat v3
);
```

Parameters

source source name whose attribute is being set

param the name of the attribute to set:
AL_POSITION
AL_VELOCITY
AL_DIRECTION

v1, v2, v3 the three ALfloat values which the attribute will be set to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is out of range.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

This function is an alternative to alSourcefv.

See Also

alSourcef, alSourcefv, alGetSourcef, alGetSource3f, alGetSourcefv

alSourcefv

Description

This function sets a floating point-vector property of a source.

```
void alSourcefv(
      ALuint source,
      ALenum param,
      ALfloat *values
);
```

Parameters

source	source name whose attribute is being set
param	the name of the attribute being set: AL_POSITION AL_VELOCITY AL_DIRECTION
values	a pointer to the vector to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is out of range.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

This function is an alternative to alSource3f.

See Also

alSourcef, alSource3f, alGetSourcef, alGetSource3f, alGetSourcefv

alSourcei

Description

This function sets an integer property of a source.

```
void alSourcei(
      ALuint source,
      ALenum param,
      ALint value
);
```

Parameters

source	source name whose attribute is being set
param	the name of the attribute to set: AL_SOURCE_RELATIVE AL_CONE_INNER_ANGLE AL_CONE_OUTER_ANGLE AL_LOOPING AL_BUFFER AL_SOURCE_STATE
value	the value to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is out of range.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The buffer name zero is reserved as a “NULL Buffer" and is accepted by *alSourcei(…, AL_BUFFER, …)* as a valid buffer of zero length. The NULL Buffer is extremely useful for detaching buffers from a source which were attached using this call or with alSourceQueueBuffers.

See Also

alSource3i, alSourceiv, alGetSourcei, alGetSource3i, alGetSourceiv

alSource3i

Description

This function sets an integer property of a source.

```
void alSourcei(
      ALuint source,
      ALenum param,
      ALint v1,
      ALint v2,
      ALint v3
);
```

Parameters

source	source name whose attribute is being set
param	the name of the attribute to set: AL_POSITION AL_VELOCITY AL_DIRECTION
v1, v2, v3	the values to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is out of range.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alSourcei, alSourceiv, alGetSourcei, alGetSource3i, alGetSourceiv

alSourceiv

Description

This function sets an integer property of a source.

```
void alSourceiv(
      ALuint source,
      ALenum param,
      ALint *values
);
```

Parameters

source	source name whose attribute is being set
param	the name of the attribute to set: AL_POSITION AL_VELOCITY AL_DIRECTION
values	the values to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is out of range.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alSourcei, alSource3i, alGetSourcei, alGetSource3i, alGetSourceiv

alGetSourcef

Description

This function retrieves a floating point property of a source.

```
void alGetSourcef(
      ALuint source,
      ALenum param,
      ALfloat *value
);
```

Parameters

source source name whose attribute is being retrieved

param the name of the attribute to retrieve:
AL_PITCH
AL_GAIN
AL_MIN_GAIN
AL_MAX_GAIN
AL_MAX_DISTANCE
AL_ROLLOFF_FACTOR
AL_CONE_OUTER_GAIN
AL_CONE_INNER_ANGLE
AL_CONE_OUTER_ANGLE
AL_REFERENCE_DISTANCE

value a pointer to the floating point value being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alSourcef, alSource3f, alSourcefv, alGetSource3f, alGetSourcefv

alGetSource3f

Description

This function retrieves three floating point values representing a property of a source.

```
void alGetSource3f(
      ALuint source,
      ALenum param,
      ALfloat *v1,
      ALfloat *v2,
      ALfloat *v3
);
```

Parameters

source	source name whose attribute is being retrieved
param	the name of the attribute being retrieved: AL_POSITION AL_VELOCITY AL_DIRECTION
v1, v2, v3	pointers to the values to retrieve

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alSourcef, alSource3f, alSourcefv, alGetSourcef, alGetSourcefv

alGetSourcefv

Description

This function retrieves a floating point-vector property of a source.

```
void alGetSourcefv(
      ALuint source,
      ALenum param,
      ALfloat *values
);
```

Parameters

source	source name whose attribute is being retrieved
param	the name of the attribute being retrieved: AL_POSITION AL_VELOCITY AL_DIRECTION
values	a pointer to the vector to retrieve

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alSourcef, alSource3f, alSourcefv, alGetSourcef, alGetSource3f

alGetSourcei

Description

This function retrieves an integer property of a source.

```
void alGetSourcei(
      ALuint source,
      ALenum pname,
      ALint *value
);
```

Parameters

source	source name whose attribute is being retrieved
pname	the name of the attribute to retrieve: AL_SOURCE_RELATIVE AL_BUFFER AL_SOURCE_STATE AL_BUFFERS_QUEUED AL_BUFFERS_PROCESSED
value	a pointer to the integer value being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alSourcei, alSource3i, alSourceiv, alGetSource3i, alGetSourceiv

alGetSource3i

Description

This function retrieves an integer property of a source.

```
void alGetSource3i(
      ALuint source,
      ALenum param,
      ALint *v1,
      ALint *v2,
      ALint *v3
);
```

Parameters

source	source name whose attribute is being retrieved
pname	the name of the attribute to retrieve: AL_POSITION AL_VELOCITY AL_DIRECTION
v1, v2, v3	pointers to the integer values being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alSourcei, alSource3i, alSourceiv, alGetSourcei, alGetSourceiv

alGetSourceiv

Description

This function retrieves an integer property of a source.

```
void alGetSourceiv(
      ALuint source,
      ALenum param,
      ALint *values
);
```

Parameters

source	source name whose attribute is being retrieved
param	the name of the attribute to retrieve: AL_POSITION AL_VELOCITY AL_DIRECTION
values	pointer to the integer values being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alSourcei, alSource3i, alSourceiv, alGetSourcei, alGetSource3i

alSourcePlay

Description

This function plays a source.

```
void alSourcePlay(
      ALuint source
);
```

Parameters

source the name of the source to be played

Possible Error States

State	*Description*
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The playing source will have its state changed to AL_PLAYING. When called on a source which is already playing, the source will restart at the beginning. When the attached buffer(s) are done playing, the source will progress to the AL_STOPPED state.

See Also

alSourcePlayv, alSourcePause, alSourcePausev, alSourceRewind, alSourceRewindv, alSourceStop, alSourceStopv

alSourcePlayv

Description

This function plays a set of sources.

```
void alSourcePlayv(
      ALsizei n,
      ALuint *sources
);
```

Parameters

n	the number of sources to be played
sources	a pointer to an array of sources to be played

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The playing sources will have their state changed to AL_PLAYING. When called on a source which is already playing, the source will restart at the beginning. When the attached buffer(s) are done playing, the source will progress to the AL_STOPPED state.

See Also

alSourcePlay, alSourcePause, alSourcePausev, alSourceRewind, alSourceRewindv, alSourceStop, alSourceStopv

alSourcePause

Description

This function pauses a source.

```
void alSourcePause(
      ALuint source
);
```

Parameters

source the name of the source to be paused

Possible Error States

State	*Description*
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The paused source will have its state changed to AL_PAUSED.

See Also

alSourcePlay, alSourcePlayv, alSourcePausev, alSourceRewind, alSourceRewindv, alSourceStop, alSourceStopv

alSourcePausev

Description

This function pauses a set of sources.

```
void alSourcePausev(
      ALsizei n,
      ALuint *sources
);
```

Parameters

n	the number of sources to be paused
sources	a pointer to an array of sources to be paused

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The paused sources will have their state changed to AL_PAUSED.

See Also

alSourcePlay, alSourcePlayv, alSourcePause, alSourceRewind, alSourceRewindv, alSourceStop, alSourceStopv

alSourceStop

Description

This function stops a source.

```
void alSourceStop(
      ALuint source
);
```

Parameters

source the name of the source to be stopped

Possible Error States

State	*Description*
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The stopped source will have its state changed to AL_STOPPED.

See Also

alSourcePlay, alSourcePlayv, alSourcePause, alSourcePausev, alSourceRewind, alSourceRewindv, alSourceStopv

alSourceStopv

Description

This function stops a set of sources.

```
void alSourceStopv(
      ALsizei n,
      ALuint *sources
);
```

Parameters

n	the number of sources to stop
sources	a pointer to an array of sources to be stopped

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The stopped sources will have their state changed to AL_STOPPED.

See Also

alSourcePlay, alSourcePlayv, alSourcePause, alSourcePausev, alSourceRewind, alSourceRewindv, alSourceStop

alSourceRewind

Description

This function stops the source and sets its state to AL_INITIAL.

```
void alSourceRewind(
      ALuint source
);
```

Parameters

source the name of the source to be rewound

Possible Error States

State	*Description*
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alSourcePlay, alSourcePlayv, alSourcePause, alSourcePausev, alSourceRewindv, alSourceStop, alSourceStopv

alSourceRewindv

Description

This function stops a set of sources and sets all their states to AL_INITIAL.

```
void alSourceRewindv(
      ALsizei n,
      ALuint *sources
);
```

Parameters

n	the number of sources to be rewound
sources	*a* pointer to an array of sources to be rewound

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alSourcePlay, alSourcePlayv, alSourcePause, alSourcePausev, alSourceRewind, alSourceStop, alSourceStopv

alSourceQueueBuffers

Description

This function queues a set of buffers on a source. All buffers attached to a source will be played in sequence, and the number of processed buffers can be detected using an alSourcei call to retrieve AL_BUFFERS_PROCESSED.

```
void alSourceQueueBuffers(
      ALuint source,
      ALsizei n,
      ALuint* buffers
);
```

Parameters

source	the name of the source to queue buffers onto
n	the number of buffers to be queued
buffers	a pointer to an array of buffer names to be queued

Possible Error States

State	***Description***
AL_INVALID_NAME	At least one specified buffer name is not valid, or the specified source name is not valid.
AL_INVALID_OPERATION	There is no current context, an attempt was made to add a new buffer which is not the same format as the buffers already in the queue, or the source already has a static buffer attached.

Version Requirements

OpenAL 1.0 or higher

Remarks

When first created, a source will be of type AL_UNDETERMINED. A successful alSourceQueueBuffers call will change the source type to AL_STREAMING.

See Also

alSourceUnqueueBuffers

alSourceUnqueueBuffers

Description

This function unqueues a set of buffers attached to a source. The number of processed buffers can be detected using an alSourcei call to retrieve AL_BUFFERS_PROCESSED, which is the maximum number of buffers that can be unqueued using this call.

```
void alSourceUnqueueBuffers(
      ALuint source,
      ALsizei n,
      ALuint* buffers
);
```

Parameters

source	the name of the source to unqueue buffers from
n	the number of buffers to be unqueued
buffers	a pointer to an array of buffer names that were removed

Possible Error States

State	***Description***
AL_INVALID_VALUE	At least one buffer can not be unqueued because it has not been processed yet.
AL_INVALID_NAME	The specified source name is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The unqueue operation will only take place if all n buffers can be removed from the queue.

See Also

alSourceQueueBuffers

Listener Functions

Properties

Property	Data Type	Description
AL_GAIN	f, fv	“master gain” value should be positive
AL_POSITION	fv, 3f, iv, 3i	X, Y, Z position
AL_VELOCITY	fv, 3f, iv, 3i	velocity vector
AL_ORIENTATION	fv, iv	orientation expressed as “at” and “up” vectors

Functions

alListenerf
alListener3f
alListenerfv
alListeneri
alListener3i
alListeneriv
alGetListenerf
alGetListener3f
alGetListenerfv
alGetListeneri
alGetListener3i
alGetListeneriv

alListenerf

Description

This function sets a floating point property for the listener.

```
void alListenerf(
      ALenum param,
      ALfloat value
);
```

Parameters:

param the name of the attribute to be set:
AL_GAIN

value the ALfloat value to set the attribute to

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alListener3f, alListenerfv, alGetListenerf, alGetListener3f, alGetListenerfv

alListener3f

Description

This function sets a floating point property for the listener.

```
void alListener3f(
      ALenum param,
      ALfloat v1,
      ALfloat v2,
      ALfloat v3
);
```

Parameters

param	the name of the attribute to set: AL_POSITION AL_VELOCITY
v1, v2, v3	the value to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alListenerf, alListenerfv, alGetListenerf, alGetListener3f, alGetListenerfv

alListenerfv

Description

This function sets a floating point-vector property of the listener.

```
void alListenerfv(
      ALenum param,
      ALfloat *values
);
```

Parameters

param	the name of the attribute to be set: AL_POSITION AL_VELOCITY AL_ORIENTATION
values	pointer to floating point-vector values

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alListenerf, alListener3f, alGetListenerf, alGetListener3f, alGetListenerfv

alListeneri

Description

This function sets an integer property of the listener.

```
void alListeneri(
      ALenum param,
      ALint value
);
```

Parameters

param	the name of the attribute to be set
value	the integer value to set the attribute to

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

There are no integer listener attributes defined for OpenAL 1.1, but this function may be used by an extension.

See Also

alListener3i, alListeneriv, alGetListeneri, alGetListener3i, alGetListeneriv

alListener3i

Description

This function sets an integer property of the listener.

```
void alListener3i(
      ALenum param,
      ALint v1,
      ALint v2,
      ALint v3
);
```

Parameters

param	the name of the attribute to be set: AL_POSITION AL_VELOCITY
v1, v2, v3	the integer values to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alListeneri, alListeneriv, alGetListeneri, alGetListener3i, alGetListeneriv

alListeneriv

Description

This function sets an integer property of the listener.

```
void alListeneriv(
      ALenum param,
      ALint *values
);
```

Parameters

param the name of the attribute to be set
AL_POSITION
AL_VELOCITY
AL_ORIENTATION

values pointer to the integer values to set the attribute to

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value given is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alListeneri, alListener3i, alGetListeneri, alGetListener3i, alGetListeneriv

alGetListenerf

Description

This function retrieves a floating point property of the listener.

```
void alGetListenerf(
      ALenum param,
      ALfloat *value
);
```

Parameters

param	the name of the attribute to be retrieved: AL_GAIN
value	a pointer to the floating point value being retrieved

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alListenerf, alListener3f, alListenerfv, alGetListener3f, alGetListenerfv

alGetListener3f

Description

This function retrieves a set of three floating point values from a property of the listener.

```
void alGetListener3f(
      ALenum param,
      ALfloat *v1,
      ALfloat *v2,
      ALfloat *v3
);
```

Parameters

param	the name of the attribute to be retrieved AL_POSITION AL_VELOCITY
v1, v2, v3	pointers to the three floating point being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alListenerf, alListener3f, alListenerfv, alGetListenerf, alGetListenerfv

alGetListenerfv

Description

This function retrieves a floating point-vector property of the listener.

```
void alGetListenerfv(
      ALenum param,
      ALfloat *values
);
```

Parameters

param	the name of the attribute to be retrieved AL_POSITION AL_VELOCITY AL_ORIENTATION
values	a pointer to the floating point-vector value being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alListenerf, alListener3f, alListenerfv, alGetListenerf, alGetListener3f

alGetListeneri

Description

This function retrieves an integer property of the listener.

```
void alGetListeneri(
      ALenum param,
      ALint *value
);
```

Parameters

param	the name of the attribute to be retrieved
value	a pointer to the integer value being retrieved

Possible Error States

State	*Description*
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

There are no integer listener attributes defined for OpenAL 1.1, but this function may be used by an extension.

See Also

alListeneri, alListener3i, alListeneriv, alGetListener3i, alGetListeneriv

alGetListener3i

Description

This function retrieves an integer property of the listener.

```
void alGetListener3i(
      ALenum param,
      ALint *v1,
      ALint *v2,
      ALint *v3
);
```

Parameters

param	the name of the attribute to be retrieved AL_POSITION AL_VELOCITY
v1, v2, v3	pointers to the integer values being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alListeneri, alListener3i, alListeneriv, alGetListeneri, alGetListeneriv

alGetListeneriv

Description

This function retrieves an integer property of the listener.

```
void alGetListeneriv(
      ALenum param,
      ALint *values
);
```

Parameters

param	the name of the attribute to be retrieved AL_POSITION AL_VELOCITY AL_ORIENTATION
values	a pointer to the integer values being retrieved

Possible Error States

State	***Description***
AL_INVALID_VALUE	The value pointer given is not valid.
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alListeneri, alListener3i, alListeneriv, alGetListeneri, alGetListener3i

State Functions

Properties

Property	Data Type	Description
AL_ DOPPLER_FACTOR	f	Global Doppler factor
AL_ SPEED_OF_SOUND	f	Speed of sound in units per second
AL_ DISTANCE_MODEL	i	Distance model enumeration value

Functions

alEnable
alDisable
alIsEnabled
alGetBoolean
alGetDouble
alGetFloat
alGetInteger
alGetBooleanv
alGetDoublev
alGetFloatv
alGetIntegerv
alGetString
alDistanceModel
alDopplerFactor
alSpeedOfSound

alEnable

Description

This function enables a feature of the OpenAL driver.

```
void alEnable(
      ALenum capability
);
```

Parameters

capability the name of a capability to enable

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified capability is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

There are no capabilities defined in OpenAL 1.1 to be used with this function, but it may be used by an extension.

See Also

alDisable, alIsEnabled

alDisable

Description

This function disables a feature of the OpenAL driver.

```
void alDisable(
      ALenum capability
);
```

Parameters

capability the name of a capability to disable

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified capability is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

There are no capabilities defined in OpenAL 1.1 to be used with this function, but it may be used by an extension.

See Also

alEnable, alIsEnabled

alIsEnabled

Description

This function returns a boolean indicating if a specific feature is enabled in the OpenAL driver.

```
ALboolean alIsEnabled(
      ALenum capability
);
```

Parameters

capability the name of a capability to enable

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified capability is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns AL_TRUE if the capability is enabled, AL_FALSE if the capability is disabled. There are no capabilities defined in OpenAL 1.1 to be used with this function, but it may be used by an extension.

See Also

alEnable, alDisable

alGetBoolean

Description

This function returns a boolean OpenAL state.

```
ALboolean alGetBoolean(
      ALenum  param
);
```

Parameters

param the state to be queried:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The boolean state described by param will be returned.

See Also

alGetBooleanv, alGetDouble, alGetDoublev, alGetFloat, alGetFloatv, alGetInteger, alGetIntegerv

alGetDouble

Description

This function returns a double precision floating point OpenAL state.

```
Aldouble alGetDouble(
      ALenum  param
);
```

Parameters

param the state to be queried:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The double value described by param will be returned.

See Also

alGetBoolean, alGetBooleanv, alGetDoublev, alGetFloat, alGetFloatv, alGetInteger, alGetIntegerv

alGetFloat

Description

This function returns a floating point OpenAL state.

```
ALfloat     alGetFloat(
      ALenum  param
);
```

Parameters

param the state to be queried:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The floating point state described by param will be returned.

See Also

alGetBoolean, alGetBooleanv, alGetDouble, alGetDoublev, alGetFloatv, alGetInteger, alGetIntegerv

alGetInteger

Description

This function returns an integer OpenAL state.

```
Alint alGetInteger(
      ALenum  param
);
```

Parameters

param the state to be queried:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The integer state described by param will be returned.

See Also

alGetBoolean, alGetBooleanv, alGetDouble, alGetDoublev, alGetFloat, alGetFloatv, alGetIntegerv

alGetBooleanv

Description

This function retrieves a boolean OpenAL state.

```
void alGetBooleanv(
      ALenum param,
      ALboolean *data
);
```

Parameters

param	the state to be returned: AL_DOPPLER_FACTOR AL_SPEED_OF_SOUND AL_DISTANCE_MODEL
data	a pointer to the location where the state will be stored

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_VALUE	The specified data pointer is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alGetBoolean, alGetDouble, alGetDoublev, alGetFloat, alGetFloatv, alGetInteger, alGetIntegerv

alGetDoublev

Description

This function retrieves a double precision floating point OpenAL state.

```
void alGetDoublev(
      ALenum param,
      ALdouble *data
);
```

Parameters

param the state to be returned:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

data a pointer to the location where the state will be stored

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_VALUE	The specified data pointer is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alGetBoolean, alGetBooleanv, alGetDouble, alGetFloat, alGetFloatv, alGetInteger, alGetIntegerv

alGetFloatv

Description

This function retrieves a floating point OpenAL state.

```
void alGetFloatv(
      ALenum param,
      ALfloat *data
);
```

Parameters

param the state to be returned:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

data a pointer to the location where the state will be stored

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_VALUE	The specified data pointer is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alGetBoolean, alGetBooleanv, alGetDouble, alGetDoublev, alGetFloat, alGetInteger, alGetIntegerv

alGetIntegerv

Description

This function retrieves an integer OpenAL state.

```
void alGetIntegerv(
      ALenum param,
      ALint *data
);
```

Parameters

param The state to be returned:
AL_DOPPLER_FACTOR
AL_SPEED_OF_SOUND
AL_DISTANCE_MODEL

data a pointer to the location where the state will be stored

Possible Error States

State	***Description***
AL_INVALID_ENUM	The specified parameter is not valid.
AL_INVALID_VALUE	The specified data pointer is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

None

See Also

alGetBoolean, alGetBooleanv, alGetDouble, alGetDoublev, alGetFloat, alGetFloatv, alGetInteger

alGetString

Description

This function retrieves an OpenAL string property.

```
const ALchar * alGetString(
      ALenum param
);
```

Parameters

param	The property to be returned AL_VENDOR AL_VERSION AL_RENDERER AL_EXTENSIONS

Possible Error States

State	*Description*
AL_INVALID_ENUM	The specified parameter is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns a pointer to a null-terminated string.

alDistanceModel

Description

This function selects the OpenAL distance model – AL_INVERSE_DISTANCE, AL_INVERSE_DISTANCE_CLAMPED, AL_LINEAR_DISTANCE, AL_LINEAR_DISTANCE_CLAMPED, AL_EXPONENT_DISTANCE, AL_EXPONENT_DISTANCE_CLAMPED, or AL_NONE.

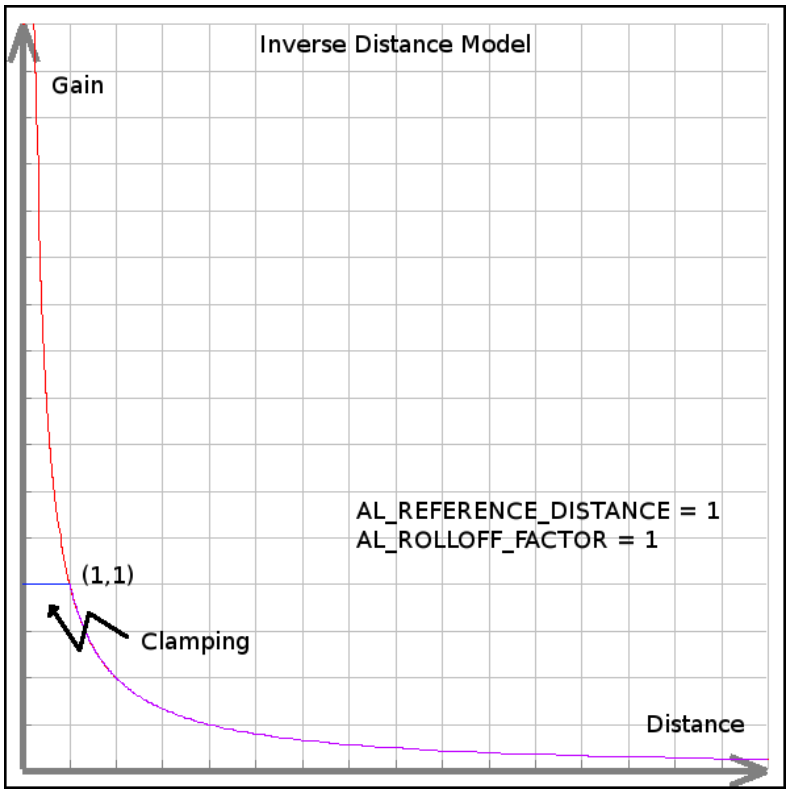
The AL_INVERSE_DISTANCE model works according to the following formula:

```
gain = AL_REFERENCE_DISTANCE / (AL_REFERENCE_DISTANCE +
            AL_ROLLOFF_FACTOR *
            (distance – AL_REFERENCE_DISTANCE));
```

The AL_INVERSE_DISTANCE_CLAMPED model works according to the following formula:

```
distance = max(distance,AL_REFERENCE_DISTANCE);
distance = min(distance,AL_MAX_DISTANCE);
gain = AL_REFERENCE_DISTANCE / (AL_REFERENCE_DISTANCE +
      AL_ROLLOFF_FACTOR *
      (distance – AL_REFERENCE_DISTANCE));
```

Here is a graph showing the inverse distance curve:



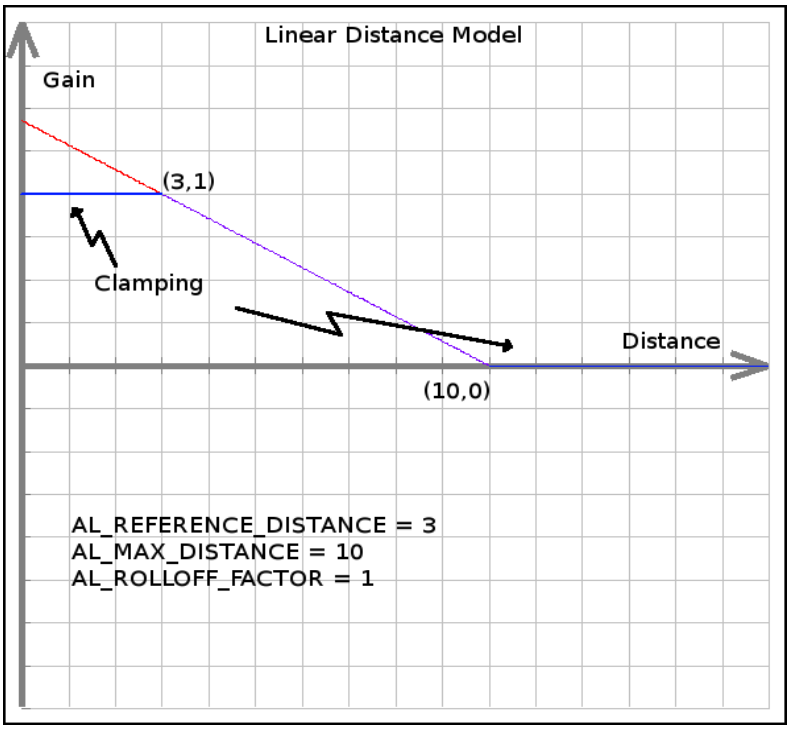
The AL_LINEAR_DISTANCE model works according to the following formula:

```
distance = min(distance, AL_MAX_DISTANCE) // avoid negative gain
gain = (1 – AL_ROLLOFF_FACTOR * (distance –
           AL_REFERENCE_DISTANCE) /
           (AL_MAX_DISTANCE – AL_REFERENCE_DISTANCE))
```

The AL_LINEAR_DISTANCE_CLAMPED model works according to the following formula:

```
distance = max(distance, AL_REFERENCE_DISTANCE)
distance = min(distance, AL_MAX_DISTANCE)
gain = (1 – AL_ROLLOFF_FACTOR * (distance –
           AL_REFERENCE_DISTANCE) /
           (AL_MAX_DISTANCE – AL_REFERENCE_DISTANCE))
```

Here is a graph showing the linear distance curve:

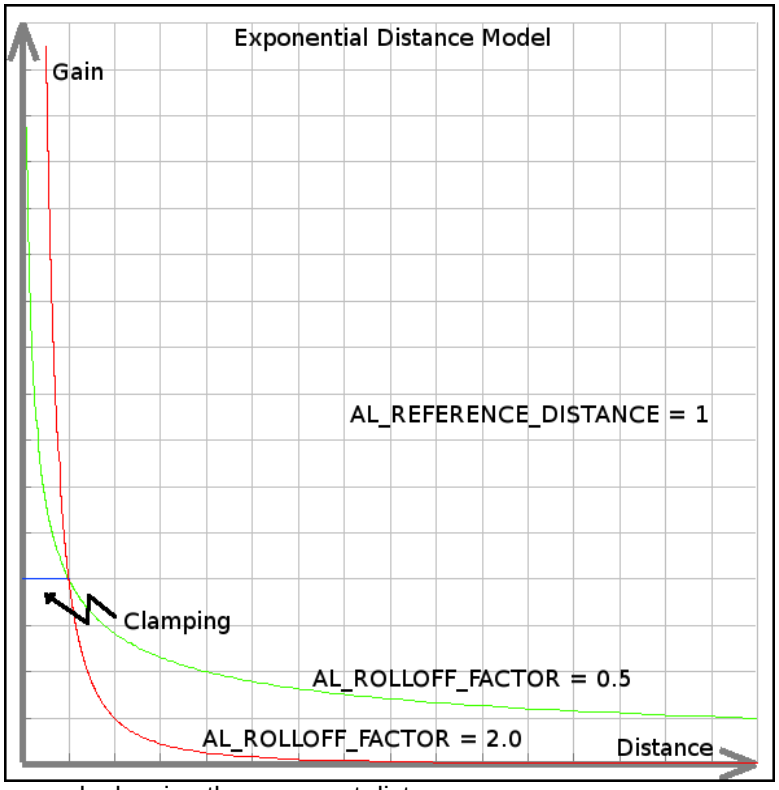


The AL_EXPONENT_DISTANCE model works according to the following formula:

```
gain = (distance / AL_REFERENCE_DISTANCE) ^
                  (- AL_ROLLOFF_FACTOR)
```

The AL_EXPONENT_DISTANCE_CLAMPED model works according to the following formula:

```
distance = max(distance, AL_REFERENCE_DISTANCE)
      distance = min(distance, AL_MAX_DISTANCE)
      gain = (distance / AL_REFERENCE_DISTANCE) ^
                  (- AL_ROLLOFF_FACTOR)
```



Here is a graph showing the exponent distance curve:

The AL_NONE model works according to the following formula:

gain = 1;

```
void alDistanceModel(
      ALenum value
);
```

Parameters

value the distance model to be set:
AL_INVERSE_DISTANCE
AL_INVERSE_DISTANCE_CLAMPED
AL_LINEAR_DISTANCE
AL_LINEAR_DISTANCE_CLAMPED
AL_EXPONENT_DISTANCE
AL_EXPONENT_DISTANCE_CLAMPED
AL_NONE

Possible Error States

State	*Description*
AL_INVALID_VALUE	The specified distance model is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The default distance model in OpenAL is AL_INVERSE_DISTANCE_CLAMPED.

alDopplerFactor

Description

This function selects the OpenAL Doppler factor value.

```
void alDopplerFactor(
      ALfloat value
);
```

Parameters

value the Doppler scale value to set

Possible Error States

State	*Description*
AL_INVALID_VALUE	The specified value is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.0 or higher

Remarks

The default Doppler factor value is 1.0.

alSpeedOfSound

Description

This function selects the speed of sound for use in Doppler calculations.

```
void alSpeedOfSound(
      ALfloat value
);
```

Parameters

value the speed of sound value to set

Possible Error States

State	*Description*
AL_INVALID_VALUE	The specified value is not valid.
AL_INVALID_OPERATION	There is no current context.

Version Requirements

OpenAL 1.1 or higher

Remarks

The default speed of sound value is 343.3.

Error Functions

Error Codes

Error Code	Description
AL_NO_ERROR	there is not currently an error
AL_INVALID_NAME	a bad name (ID) was passed to an OpenAL function
AL_INVALID_ENUM	an invalid enum value was passed to an OpenAL function
AL_INVALID_VALUE	an invalid value was passed to an OpenAL function
AL_INVALID_OPERATION	the requested operation is not valid
AL_OUT_OF_MEMORY	the requested operation resulted in OpenAL running out of memory

Functions

alGetError

alGetError

Description

This function returns the current error state and then clears the error state.

```
ALenum alGetError(ALvoid);
```

Parameters

None

Possible Error States

None

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns an Alenum representing the error state. When an OpenAL error occurs, the error state is set and will not be changed until the error state is retrieved using alGetError. Whenever alGetError is called, the error state is cleared and the last state (the current state when the call was made) is returned. To isolate error detection to a specific portion of code, alGetError should be called before the isolated section to clear the current error state.

Extension Functions

Functions

alIsExtensionPresent

Description

This function tests if a specific extension is available for the OpenAL driver.

```
ALboolean alIsExtensionPresent(
      const ALchar *extname
);
```

Parameters

extname	a null-terminated string describing the desired extension

Possible Error States

State	***Description***

alGetProcAddress

Description

This function returns the address of an OpenAL extension function.

```
void * alGetProcAddress(
      const ALchar *fname
);
```

Parameters

fname	a null-terminated string containing the function name

Possible Error States

None

Version Requirements

OpenAL 1.0 or higher

Remarks

The return value is a pointer to the specified function. The return value will be NULL if the function is not found.

See Also

alIsExtensionPresent, alGetEnumValue

alGetEnumValue

Description

This function returns the enumeration value of an OpenAL enum described by a string.

```
ALenum alGetEnumValue(
      const ALchar *ename
);
```

Parameters

ename a null-terminated string describing an OpenAL enum

Possible Error States

None

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns the actual ALenum described by a string. Returns NULL if the string doesn't describe a valid OpenAL enum.

See Also

alIsExtensionPresent, alGetProcAddress

Context Management Functions

Properties

Property	Data Type	Description
ALC_FREQUENCY	I	output frequency
ALC_MONO_SOURCES	I	requested number of mono sources
ALC_STEREO_SOURCES	I	requested number of stereo sources
ALC_REFRESH	I	update rate of context processing
ALC_SYNC	i	flag indicating a synchronous context

Functions

alcCreateContext
alcMakeContextCurrent
alcProcessContext
alcSuspendContext
alcDestroyContext
alcGetCurrentContext
alcGetContextsDevice

alcCreateContext

Description

This function creates a context using a specified device.

```
ALCcontext * alcCreateContext(
      ALCdevice *device,
      ALCint* attrlist
);
```

Parameters

device a pointer to a device

attrlist a pointer to a set of attributes:
ALC_FREQUENCY
ALC_MONO_SOURCES
ALC_REFRESH
ALC_STEREO_SOURCES
ALC_SYNC

Possible Error States

State	***Description***
ALC_INVALID_VALUE	An additional context can not be created for this device.
ALC_INVALID_DEVICE	The specified device is not a valid output device.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns a pointer to the new context (NULL on failure).

The attribute list can be NULL, or a zero terminated list of integer pairs composed of valid ALC attribute tokens and requested values.

See Also

alcDestroyContext, alcMakeContextCurrent

alcMakeContextCurrent

Description

This function makes a specified context the current context.

```
ALCboolean alcMakeContextCurrent(
      ALCcontext *context
);
```

Parameters

context a pointer to the new context

Possible Error States

State	*Description*
ALC_INVALID_CONTEXT	The specified context is invalid.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns ALC_TRUE on success, or ALC_FALSE on failure.

See Also

alcCreateContext, alcDestroyContext

alcProcessContext

Description

This function tells a context to begin processing.

```
void alcProcessContext(
      ALCcontext *context
);
```

Parameters

context a pointer to the new context

Possible Error States

State	*Description*
ALC_INVALID_CONTEXT	The specified context is invalid.

Version Requirements

OpenAL 1.0 or higher

Remarks

When a context is suspended, changes in OpenAL state will be accepted but will not be processed. alcSuspendContext can be used to suspend a context, and then all the OpenAL state changes can be applied at once, followed by a call to alcProcessContext to apply all the state changes immediately. In some cases, this procedure may be more efficient than application of properties in a non-suspended state. In some implementations, process and suspend calls are each a NOP.

See Also

alcSuspendContext

alcSuspendContext

Description

This function suspends processing on a specified context.

```
void alcSuspendContext(
      ALCcontext *context
);
```

Parameters

context a pointer to the context to be suspended

Possible Error States

State	*Description*
ALC_INVALID_CONTEXT	The specified context is invalid.

Version Requirements

OpenAL 1.0 or higher

Remarks

When a context is suspended, changes in OpenAL state will be accepted but will not be processed. A typical use of alcSuspendContext would be to suspend a context, apply all the OpenAL state changes at once, and then call alcProcessContext to apply all the state changes at once. In some cases, this procedure may be more efficient than application of properties in a non-suspended state. In some implementations, process and suspend calls are each a NOP.

See Also

alcProcessContext

alcDestroyContext

Description

This function destroys a context.

```
void alcDestroyContext(
      ALCcontext *context
);
```

Parameters

context a pointer to the new context

Possible Error States

State	*Description*
ALC_INVALID_CONTEXT	The specified context is invalid.

Version Requirements

OpenAL 1.0 or higher

Remarks

A context which is not current can be destroyed at any time (all sources within that context will also be deleted). alcMakeContextCurrent should be used to make sure the context to be destroyed is not current (NULL is valid for alcMakeContextCurrent).

See Also

alcCreateContext, alcMakeContextCurrent

alcGetCurrentContext

Description

This function retrieves the current context.

```
ALCcontext * alcGetCurrentContext( ALCvoid );
```

Parameters

None

Possible Error States

None

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns a pointer to the current context.

See Also

alcGetContextsDevice

alcGetContextsDevice

Description

This function retrieves a context's device pointer.

```
ALCdevice * alcGetContextsDevice( ALCcontext *context );
```

Parameters

context a pointer to a context

Possible Error States

State	*Description*
ALC_INVALID_CONTEXT	The specified context is invalid.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns a pointer to the specified context's device.

See Also

alcGetCurrentContext

Context Error Functions

Error Codes

Error Code	Description
ALC_NO_ERROR	there is not currently an error
ALC_INVALID_DEVICE	a bad device was passed to an OpenAL function
ALC_INVALID_CONTEXT	a bad context was passed to an OpenAL function
ALC_INVALID_ENUM	an unknown enum value was passed to an OpenAL function
ALC_INVALID_VALUE	an invalid value was passed to an OpenAL function
ALC_OUT_OF_MEMORY	the requested operation resulted in OpenAL running out of memory

Functions

alcGetError

alcGetError

Description

This function retrieves the current context error state.

```
ALCenum alcGetError( ALCdevice *device );
```

Parameters

device a pointer to the device to retrieve the error state from

Possible Error States

None

Version Requirements

OpenAL 1.0 or higher

Remarks

None

Context Device Functions

Functions

alcOpenDevice
alcCloseDevice

alcOpenDevice

Description

This function opens a device by name.

```
ALCdevice *alcOpenDevice(
      const ALCchar *devicename
);
```

Parameters

devicename a null-terminated string describing a device

Possible Error States

The return value will be NULL if there is an error.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns a pointer to the opened device. Will return NULL if a device can not be opened.

See Also

alcCloseDevice

alcCloseDevice

Description

This function closes a device by name.

```
ALCboolean alcCloseDevice(
      ALCdevice *device
);
```

Parameters

device a pointer to an opened device

Possible Error States

State	*Description*

Context Extension Functions

Functions

alcIsExtensionPresent
alcGetProcAddress
alcGetEnumValue

alcIsExtensionPresent

Description

This function queries if a specified context extension is available.

```
ALCboolean alcIsExtensionPresent(
      ALCdevice *device,
      const ALCchar *extName
);
```

Parameters

device	a pointer to the device to be queried for an extension
extName	a null-terminated string describing the extension

Possible Error States

State	*Description*
ALC_INVALID_VALUE	The string pointer is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns ALC_TRUE if the extension is available, ALC_FALSE if the extension is not available.

See Also

alcGetProcAddress, alcGetEnumValue

alcGetProcAddress

Description

This function retrieves the address of a specified context extension function.

```
void  * alcGetProcAddress(
      ALCdevice *device,
      const ALCchar *funcName
);
```

Parameters

device	a pointer to the device to be queried for the function
funcName	a null-terminated string describing the function

Possible Error States

State	*Description*
ALC_INVALID_VALUE	The string pointer is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns the address of the function, or NULL if it is not found.

See Also

alcIsExtensionPresent, alcGetEnumValue

alcGetEnumValue

Description

This function retrieves the enum value for a specified enumeration name.

```
ALCenum alcGetEnumValue(
      ALCdevice *device,
      const ALCchar *enumName
);
```

Parameters

device	a pointer to the device to be queried
enumName	a null terminated string describing the enum value

Possible Error States

State	***Description***
ALC_INVALID_VALUE	The string pointer is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

Returns the enum value described by the enumName string. This is most often used for querying an enum value for an ALC extension.

See Also

alcIsExtensionPresent, alcGetProcAddress

Context State Functions

Functions

alcGetString
alcGetIntegerv

alcGetString

Description

This function returns pointers to strings related to the context.

```
const ALCchar * alcGetString(
      ALCdevice *device,
      ALenum param
);
```

Parameters

device a pointer to the device to be queried

param an attribute to be retrieved:
ALC_DEFAULT_DEVICE_SPECIFIER
ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER
ALC_DEVICE_SPECIFIER
ALC_CAPTURE_DEVICE_SPECIFIER
ALC_EXTENSIONS

Possible Error States

State	***Description***
ALC_INVALID_ENUM	The specified parameter is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

ALC_DEFAULT_DEVICE_SPECIFIER will return the name of the default output device.

ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER will return the name of the default capture device.

ALC_DEVICE_SPECIFIER will return the name of the specified output device if a pointer is supplied, or will return a list of all available devices if a NULL device pointer is supplied. A list is a pointer to a series of strings separated by NULL characters, with the list terminated by two NULL characters. See Enumeration Extension for more details.

ALC_CAPTURE_DEVICE_SPECIFIER will return the name of the specified capture device if a pointer is supplied, or will return a list of all available devices if a NULL device pointer is supplied.

ALC_EXTENSIONS returns a list of available context extensions, with each extension separated by a space and the list terminated by a NULL character.

alcGetIntegerv

Description

This function returns integers related to the context.

```
void alcGetIntegerv(
      ALCdevice *device,
      ALCenum param,
      ALCsizei size,
      ALCint *data
);
```

Parameters

device — a pointer to the device to be queried

param — an attribute to be retrieved:
ALC_MAJOR_VERSION
ALC_MINOR_VERSION
ALC_ATTRIBUTES_SIZE
ALC_ALL_ATTRIBUTES

size — the size of the destination buffer provided

data — a pointer to the data to be returned

Possible Error States

State	***Description***
ALC_INVALID_VALUE	The specified data pointer or size is not valid.
ALC_INVALID_ENUM	The specified parameter is not valid.
ALC_INVALID_DEVICE	The specified device is not valid.
ALC_INVALID_CONTEXT	The specified context is not valid.

Version Requirements

OpenAL 1.0 or higher

Remarks

The versions returned refer to the specification version that the implementation meets.

Context Capture Functions

Functions

alcCaptureOpenDevice

Description

This function opens a capture device by name.

```
ALCdevice * alcCaptureOpenDevice(
      const ALCchar *devicename,
      ALCuint frequency,
      ALCenum format,
      ALCsizei buffersize
);
```

Parameters

devicename	a pointer to a device name string
frequency	the frequency that the data should be captured at
format	the requested capture buffer format
buffersize	the size of the capture buffer

Possible Error States

State	***Description***
ALC_INVALID_VALUE	One of the parameters has an invalid value.
ALC_OUT_OF_MEMORY	The specified device is invalid, or can not capture audio.

Version Requirements

OpenAL 1.1 or higher

Remarks

Returns the capture device pointer, or NULL on failure.

See Also

alcCaptureCloseDevice

alcCaptureCloseDevice

Description

This function closes the specified capture device.

```
ALCboolean alcCaptureCloseDevice(
      ALCdevice *device
);
```

Parameters

device a pointer to a capture device

Possible Error States

State	*Description*
ALC_INVALID_DEVICE	The specified device is not a valid capture device.

Version Requirements

OpenAL 1.1 or higher

Remarks

Returns ALC_TRUE if the close operation was successful, ALC_FALSE on failure.

See Also

alcCaptureOpenDevice

alcCaptureStart

Description

This function begins a capture operation.

```
void alcCaptureStart(
      ALCdevice *device
);
```

Parameters

device a pointer to a capture device

Possible Error States

State	*Description*
ALC_INVALID_DEVICE	The specified device is not a valid capture device.

Version Requirements

OpenAL 1.1 or higher

Remarks

alcCaptureStart will begin recording to an internal ring buffer of the size specified when opening the capture device. The application can then retrieve the number of samples currently available using the ALC_CAPTURE_SAPMPLES token with alcGetIntegerv. When the application determines that enough samples are available for processing, then it can obtain them with a call to alcCaptureSamples.

See Also

alcCaptureStop, alcCaptureSamples

alcCaptureStop

Description

This function stops a capture operation.

```
void alcCaptureStop(
      ALCdevice *device
);
```

Parameters

device a pointer to a capture device

Possible Error States

State	*Description*
ALC_INVALID_DEVICE	The specified device is not a valid capture device.

Version Requirements

OpenAL 1.1 or higher

Remarks

None

See Also

alcCaptureStart, alcCaptureSamples

alcCaptureSamples

Description

This function completes a capture operation, and does not block.

```
void alcCaptureSamples(
      ALCdevice *device,
      ALCvoid *buffer,
      ALCsizei samples
);
```

Parameters

device	a pointer to a capture device
buffer	a pointer to a data buffer, which must be large enough to accommodate *samples* number of samples
samples	the number of samples to be retrieved

Possible Error States

State	***Description***

ALC and AL Function Lists

Functions new to OpenAL 1.1 are *italicized* and **boldface**.

ALC Functions

```
alcCreateContext
alcMakeContextCurrent
alcProcessContext
alcSuspendContext
alcDestroyContext
alcGetCurrentContext
alcGetContextsDevice
alcOpenDevice
alcCloseDevice
alcGetError
alcIsExtensionPresent
alcGetProcAddress
alcGetEnumValue
alcGetString
alcGetIntegerv
alcCaptureOpenDevice
alcCaptureCloseDevice
alcCaptureStart
alcCaptureStop
alcCaptureSamples
```

AL Functions

```
alEnable
alDisable
alIsEnabled
alGetString
alGetBooleanv
alGetIntegerv
alGetFloatv
alGetDoublev
alGetBoolean
alGetInteger
alGetFloat
alGetDouble
alGetError
alIsExtensionPresent
alGetProcAddress
alGetEnumValue
alListenerf
alListener3f
alListenerfv
alListeneri
alListener3i
alListeneriv
alGetListenerf
alGetListener3f
```

alGetListenerfv
alGetListeneri
alGetListener3i
alGetListeneriv
alGenSources
alDeleteSources
alIsSource
alSourcef
alSource3f
alSourcefv
alSourcei
alSource3i
alSourceiv
alGetSourcef
alGetSource3f
alGetSourcefv
alGetSourcei
alGetSource3i
alGetSourceiv
alSourcePlayv
alSourceStopv
alSourceRewindv
alSourcePausev
alSourcePlay
alSourceStop
alSourceRewind
alSourcePause
alSourceQueueBuffers
alSourceUnqueueBuffers
alGenBuffers
alDeleteBuffers
alIsBuffer
alBufferf
alBuffer3f
alBufferfv
alBufferi
alBuffer3i
alBufferiv
alGetBufferf
alGetBuffer3f
alGetBufferfv
alGetBufferi
alGetBuffer3i
alGetBufferiv
alDopplerFactor
alDopplerVelocity
alSpeedOfSound
alDistanceModel

Standard Extensions to OpenAL

The Enumeration Extension enables a developer to retrieve a list of device strings, identifying the OpenAL devices present on a system. This allows an application to present the user with a choice of valid rendering devices at run-time.

Enumeration Extension

The Enumeration Extension enables the application developer to retrieve a list of device strings identifying the different OpenAL rendering and capture devices present on the user's PC. The OpenAL router takes care of querying the user's system to find valid device implementations. Any of the strings returned by the enumeration extension can be used to create a device during initialization via alcOpenDevice. This extension is critical if you want to enable the user to select at run-time which device should be used to render your OpenAL audio.

Naturally device enumeration is a very platform-specific topic. The mechanism might not be implemented on platforms such as games consoles with fixed capabilities, where multiple rendering devices are unnecessary.

Note that on PC the standard Enumeration Extension will not identify every potential OpenAL output path. It will not return all the possible outputs in situations where the user has more than one audio device installed, or under Windows Vista where the audio system specifies different "endpoints" for sound such as Speakers, S/PDIF, etc... If you require complete control over the choice of output path, use the "Enumerate All" extension.

For full details on making use of the different devices you might come across on the Windows PC platform, see the accompanying OpenAL Deployment Guide (PC Windows).

Detecting the Enumeration Extension

To check whether the OpenAL libraries expose the Enumeration extension, use the OpenAL function call alcIsExtensionPresent and the name "ALC_ENUMERATION_EXT".

```
if (alcIsExtensionPresent(NULL, "ALC_ENUMERATION_EXT") ==
AL_TRUE)
      // Enumeration Extension Found
```

Retrieving device names

If the extension is found, the developer can retrieve a string containing NULL-separated device name strings (the list is terminated with two consecutive NULL characters), and a string containing the name of the default device.

To retrieve the string listing all the devices present, the developer should use the OpenAL function call alcGetString with the name "ALC_DEVICE_SPECIFIER".

To retrieve the string containing the name of the default device, the developer should use the OpenAL function call alcGetString with the name "ALC_DEFAULT_DEVICE_SPECIFIER".

```
const ALCchar *devices;
const ALCchar *defaultDeviceName;

// Pass in NULL device handle to get list of devices
devices = alcGetString(NULL, ALC_DEVICE_SPECIFIER);
// devices contains the device names, separated by NULL
// and terminated by two consecutive NULLs.

defaultDeviceName = alcGetString(NULL,
ALC_DEFAULT_DEVICE_SPECIFIER);
// defaultDeviceName contains the name of the default
// device
```

Parsing the device string

It is trivial to parse the device string and retrieve the names of the individual devices. Ideally these will be presented to the user in the application configuration GUI, to enable the user to select the desired device at initialization time.

Checking the current device name

The developer can check to see the name of the device that was actually opened using the function call alcGetString with a pointer to an open device and the name "ALC_DEVICE_SPECIFIER".

```
ALCdevice *pMyDevice;
const ALCchar *actualDeviceName;

// Open the default device
pMyDevice=alcOpenDevice(NULL)

// Pass in valid device pointer to get the name of the open
// device

actualDeviceName = alcGetString(pMyDevice, ALC_DEVICE_SPECIFIER);
// actualDeviceName contains the name of the open device
```

Enumeration Names

ALC_ENUMERATION_EXT

Use with alcIsExtensionPresent to detect if the enumeration extension is available.

ALC_DEVICE_SPECIFIER

Use with alcGetString and a NULL device pointer to retrieve a string containing the available device names, separated with NULL characters and terminated by two consecutive NULL characters.

Use with alcGetString and a pointer to a previously-opened device to ascertain the device's name.

ALC_CAPTURE_DEVICE_SPECIFIER

Use with alcGetString and a NULL device pointer to retrieve a string containing the available capture device names, separated with NULL characters and terminated by two consecutive NULL characters.

Use with alcGetString and a pointer to a previously-opened capture device to ascertain the device's name.

ALC_DEFAULT_DEVICE_SPECIFIER

Use with alcGetString with a NULL Device identifier to retrieve a NULL-terminated string containing the name of the default device.

ALC_CAPTURE_DEFAULT_DEVICE_SPECIFIER

Use with alcGetString with a NULL Device identifier to retrieve a NULL-terminated string containing the name of the default capture device.

Creative Labs' Extensions to OpenAL

Creative has introduced a number of extensions to OpenAL, many of which take advantage of the unique features of their soundcards. The "Enumerate All" extension is similar to the Core OpenAL "Enumeration Extension" but is extended to cover all available soundcards (and audio end-points in Windows Vista). The "X-RAM" extension allows a developer to utilize on-board audio RAM for storing OpenAL buffers. The "Multi-Channel Buffers" extension allows a developer to play multi-channel buffers (e.g. 5.1). Finally, the generic "Effects Extension (EFX)" allows an application to use effects such as reverb and low-pass filters to create realistic 3D aural worlds.

Enumerate All Extension

The Enumerate All Extension enables the application developer to retrieve a complete list of device strings identifying all the available OpenAL rendering devices and paths present on the user's PC. It works in exactly the same manner as the Enumeration Extension, but it detects additional audio paths that the standard extension will ignore. For instance, it will return all the possible outputs in situations where the user has more than one audio device installed, or under Windows Vista where the audio system specifies different "endpoints" for sound such as Speakers, S/PDIF, etc... If you don't require such complete control over the choice of output path, then use the standard Enumeration Extension.

Detecting the Enumerate All Extension

To check whether the OpenAL libraries expose the Enumerate All extension, use the OpenAL function call alcIsExtensionPresent and the name "ALC_ENUMERATE_ALL_EXT".

```
if (alcIsExtensionPresent(NULL, "ALC_ENUMERATE_ALL_EXT") ==
AL_TRUE)
      // Enumerate All Extension Found
```

Retrieving device names

If the extension is found, the developer can retrieve a string containing NULL-separated device name strings (the list is terminated with two consecutive NULL characters), and a string containing the name of the default device.

To retrieve the string listing all the devices present, the developer should use the OpenAL function call alcGetString with the name "ALC_ALL_DEVICES_SPECIFIER".

To retrieve the string containing the name of the default device, the developer should use the OpenAL function call alcGetString with the name "ALC_DEFAULT_ALL_DEVICES_SPECIFIER".

```
const ALCchar *devices;
const ALCchar *defaultDeviceName;

// Pass in NULL device handle to get list of *all* devices
devices = alcGetString(NULL, ALC_ALL_DEVICES_SPECIFIER);
// devices contains *all* the device names, separated by NULL
// and terminated by two consecutive NULLs.

defaultDeviceName = alcGetString(NULL,
ALC_DEFAULT_ALL_DEVICES_SPECIFIER);
// defaultDeviceName contains the name of the default
// device
```

Any of the strings returned by the Enumerate All extension can be used to create a device during initialization via alcOpenDevice.

Enumeration Names

ALC_ENUMERATE_ALL_EXT

Use with alcIsExtensionPresent to detect if the Enumerate All extension is available.

ALC_ALL_DEVICES_SPECIFIER

Use with alcGetString and a NULL device pointer to retrieve a string containing the names of all available devices and audio output paths, separated with NULL characters and terminated by two consecutive NULL characters.

ALC_DEFAULT_ALL_DEVICES_SPECIFIER

Use with alcGetString with a NULL Device identifier to retrieve a NULL-terminated string containing the name of the default device.

X-RAM

With the introduction of the Sound Blaster X-Fi™ series of audio cards, Creative has launched a range of products that include on-board RAM. 'X-RAM' is provided on the top-end Sound Blaster X-Fi solutions (Sound Blaster X-Fi Fatal1ty™ FPS and Sound Blaster X-Fi Elite Pro). These products feature 64MB of X-RAM that can only be used for audio purposes. With the availability of X-RAM, developers can now improve performance issues related to playing audio in their applications and increase the overall quality of their sound when X-RAM is available.

X-RAM Usage Scenarios

Detecting the presence of X-RAM offers new possibilities to application developers. As a fixed resource dedicated to storing audio samples, an application can use X-RAM to improve the performance and quality of an application.

When X-RAM could be used: -

Improving Quality

An application that detects X-RAM can use higher quality audio assets that it might not be able to use otherwise.

Improving Performance

A game that detects X-RAM can decompress compressed audio samples at load time into the X-RAM so that the application does not have to spend precious processor cycles decompressing data during runtime.

When X-RAM should not be used: -

Streaming

There is an overhead involved with uploading data to the memory which means that X-RAM is not recommended for storing AL Buffers, whose contents will be constantly changing, e.g. when queuing buffers on an Open AL Source.

X-RAM Modes

The X-RAM extension to Open AL has two modes of operation – an 'automatic' mode (the default) and a 'managed' mode. In automatic mode an application does not need to make any function calls, or even query for any extensions, and Open AL buffers will automatically be loaded into X-RAM if it is found and has enough storage space. In managed mode the application developer has complete control over which Open AL Buffers are uploaded to X-RAM or not. Modes are set on individual Open AL Buffers and *must* be set before audio data is copied to the buffer. Attempts to change the Mode on a buffer that already has audio data will fail.

Automatic Mode (AL_STORAGE _AUTOMATIC)

The default buffer mode allows legacy applications to take advantage of the on-board memory. In automatic mode, the first call to alBufferData after a Buffer has been generated, will attempt to allocate the memory in X-RAM. If there is not enough memory available then an attempt to allocate system memory is made. If there is not enough system memory then the AL error AL_OUT_OF_MEMORY will be set as per the OpenAL 1.0 specification.

If a future alBufferData call is made on a buffer in automatic mode, the driver will assume that the application is using the AL Buffer for streaming (requiring regular updates to the audio data in the buffer), and the sample data will be moved from X-RAM to host memory. If there is not enough system memory then the AL error AL_OUT_OF_MEMORY will be set as per the OpenAL 1.0 specification.

Manual Mode - Hardware (AL_STORAGE _HARDWARE)

In hardware mode a buffer will be uploaded to X-RAM. A buffer in this mode is expected to be used as a single shot or looping sound, but can be reloaded if desired.

If an alBufferData call is made on a buffer in hardware mode an attempt to allocate X-RAM storage for the buffer data is made. If there is not enough X-RAM then the AL error AL_OUT_OF_MEMORY will be set as per the OpenAL 1.0 specification.

Manual Mode – Accessible (AL_STORAGE _ACCESSIBLE)

In accessible mode a buffer is to be placed where the overhead of loading the buffer is minimal. Currently this is assumed to be system memory but in future products, with potentially faster busses, the buffer will be allocated wherever is most applicable. When a buffer is put in this mode it is expected that it will be reloaded numerous times as in a streaming situation.

If an alBufferData call is made on a buffer in accessible mode an attempt to allocate system memory is always made. If there is not enough system memory then the AL error AL_OUT_OF_MEMORY should be set as per the OpenAL 1.0 specification.

Detecting X-RAM

To query for the presence of an audio card with X-RAM, use the Open AL alIsExtensionPresent function call and the name “EAX-RAM”.

```
if (alIsExtensionPresent("EAX-RAM") == AL_TRUE)
      // X-RAM Found
```

If the extension is found, an application that wishes to change Buffer Modes should query for the X-RAM extension functions using alGetProcAddress.

```
EAXSetBufferMode g_eaxSetMode;
EAXGetBufferMode g_eaxGetMode;


g_eaxSetMode = (EAXSetBufferMode)
      alGetProcAddress("EAXSetBufferMode");
g_eaxGetMode = (EAXGetBufferMode)
      alGetProcAddress("EAXGetBufferMode");
```

The EAXSetBufferMode and EAXGetBufferMode function definitions are defined in xram.h.

The final step in preparing an application to use X-RAM functionality is to query for the values of the X-RAM enumerations using alGetEnumValue. AL_EAX_RAM_SIZE and AL_EAX_RAM_FREE are used with alGetInteger to retrieve the total amount of X-RAM and the amount of free X-RAM. AL_STORAGE_AUTOMATIC, AL_STORAGE_HARDWARE and AL_STORAGE_ACCESSIBLE are used with the EAXSetBufferMode and EAXGetBufferMode functions.

```
ALenum g_eXRAMSize, g_eXRAMFree;
ALenum g_eXRAMAuto, g_eXRAMHardware, g_eXRAMAccessible;

g_eXRAMSize = alGetEnumValue("AL_EAX_RAM_SIZE");
g_eXRAMFree = alGetEnumValue("AL_EAX_RAM_FREE");
g_eXRAMAuto = alGetEnumValue("AL_STORAGE_AUTOMATIC");
g_eXRAMHardware = alGetEnumValue("AL_STORAGE_HARDWARE");
g_eXRAMAccessible = alGetEnumValue("AL_STORAGE_ACCESSIBLE");
```

To query for the total amount or available X-RAM on the soundcard, an application can use the alGetInteger function with the AL_EAX_RAM_SIZE and AL_EAX_RAM_FREE enum values.

```
ALint iRAMSizeMB;
ALint iRAMFreeMB;

iRAMSizeMB = alGetInteger(g_eXRAMSize) / (1024*1024);
iRAMFreeMB = alGetInteger(g_eXRAMFree) / (1024*1024);
```

EAXSetBufferMode

The **EAXSetBufferMode** function is used to set the storage Mode of an array of Open AL Buffers.

```
ALboolean EAXSetBufferMode(
      ALsizei n,
      ALuint *buffers,
      ALint value
);
```

Parameters

n
The number of Open AL Buffers pointed to by *buffers*.

buffers
An array of Open AL Buffer handles.

value
The storage mode that should be used for all the given buffers. Should be the value of one of the following enum names: -

AL_STORAGE_AUTOMATIC
AL_STORAGE_HARDWARE
AL_STORAGE_ACCESSIBLE

Return Values

AL_TRUE if all the AL Buffers were successfully set to the requested storage mode, AL_FALSE otherwise.

Remarks

None.

See Also

EAXGetBufferMode

EAXGetBufferMode

The **EAXGetBufferMode** function is used to retrieve the storage Mode of a particular Open AL Buffer.

```
ALenum EAXGetBufferMode(
      ALuint buffer,
      ALint *pReserved
);
```

Parameters

buffer
The handle of an Open AL Buffer.

pReserved
Should be set to NULL.

Return Values

The Storage Mode assigned to this Open AL Buffer. One of the following enum names: -

AL_STORAGE_AUTOMATIC
AL_STORAGE_HARDWARE
AL_STORAGE_ACCESSIBLE

Remarks

None.

See Also

EAXSetBufferMode

Enumeration Names

AL_EAX_RAM_SIZE

Use with alGetInteger to retrieve the total amount of X-RAM in bytes.

AL_EAX_RAM_FREE

Use with alGetInteger to retrieve the amount of free X-RAM in bytes.

AL_STORAGE_AUTOMATIC

See X-RAM Modes.

AL_STORAGE_HARDWARE

See X-RAM Modes.

AL_STORAGE_ACCESSIBLE

See X-RAM Modes.

Multi-Channel Buffers

The multi-channel extension provides a mechanism to play multi-channel data via OpenAL. A variety of formats are supported. Multi-channel buffers can be attached or queued on a source. Note that when using the “Generic Software” device, the multi-channel buffers are mixed down to a stereo output. On a hardware device (such as the “Generic Hardware” device or a native device), each channel of a buffer requires a hardware voice. So, for example playing a buffer using the AL_FORMAT_51CHN16 format will require 6 free hardware voices. If the hardware resources are unavailable, the call to alSourceQueueBuffers or alSourcei will fail.

Formats supported:

4 channels, 16 bit data
6 channels (5.1), 16 bit data
7 channels (6.1), 16 bit data
8 channels (7.1), 16 bit data

Before using any of the different multi-channel buffers, use alGetEnumValue to check if the format is supported.

```
ALenum      eBufferFormat = 0;
eBufferFormat = alGetEnumValue("AL_FORMAT_51CHN16");
if (!eBufferFormat)
{
      printf("No support for 5.1 playback!\n");
      return 0;
}
```

AL_FORMAT_QUAD16

This describes a 4 channels buffer of 16 bit samples.

Data organisation :

Sample 1, front left speaker
Sample 1, front right speaker

Sample 1, front center speaker
Sample 1, low frequency speaker
Sample 1, back left speaker
Sample 1, back right speaker

Then

Sample 2, front left speaker
Sample 2, front right speaker…

AL_FORMAT_61CHN16

This describes a 6.1 (7 channels) buffer of 16 bit samples.

Data organisation :

Sample 1, front left speaker
Sample 1, front right speaker
Sample 1, front center speaker
Sample 1, low frequency speaker
Sample 1, back left speaker
Sample 1, back right speaker
Sample 1, back center speaker

Then

Sample 2, front left speaker
Sample 2, front right speaker…

AL_FORMAT_71CHN16

This describes a 7.1 (8 channels) buffer of 16 bit samples.

Data organisation :

Sample 1, front left speaker
Sample 1, front right speaker
Sample 1, front center speaker
Sample 1, low frequency speaker
Sample 1, back left speaker
Sample 1, back right speaker
Sample 1, side left speaker
Sample 1, side right speaker

Then

Sample 2, front left speaker
Sample 2, front right speaker…

Effects Extension (EFX)

Information about the Effects Extension to OpenAL can be found in the “Effects Extension Guide”.