ISACT SDK Programmer’s Guide

Table of Contents

ISACT Game Audio Engine Overview

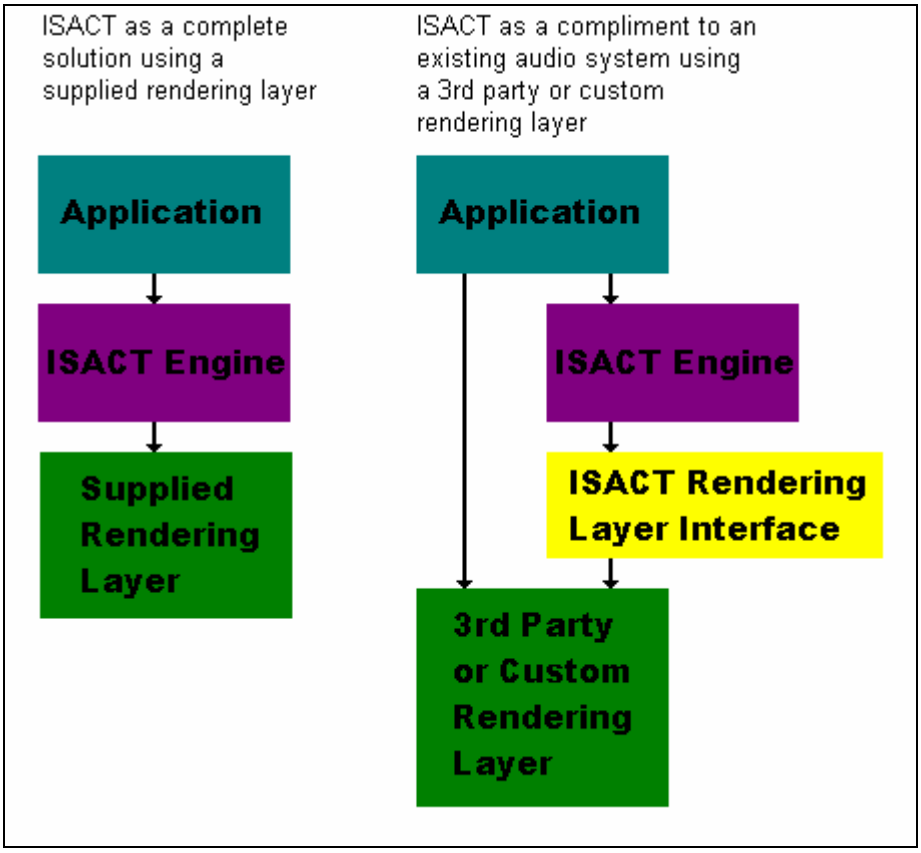
The ISACT (Interactive Spatial Audio Composition Technology) Audio Engine can be integrated into a software application for the purpose of playing back audio content created with the ISACT Production Studio (IPS) software. Content created with IPS is loaded into the ISACT runtime engine for playback by an application.

ISACT Game Audio Engine Architecture

The ISACT engine is a high level component that interprets the content created with IPS into a low level set of audio rendering commands. ISACT then uses a rendering layer to render the final audio. The ISACT SDK comes with rendering layers for most of the supported platforms. Alternatively other rendering layers can be used. For information on creating your own ISACT rendering layer see the ISACT Rendering Layer SDK.

Separating the ISACT engine from the rendering layer allows it to be integrated with many APIs and/or 3rd party solutions. A rendering layer could be created from an applications existing audio rendering system allowing ISACT to become either a complete solution or a subset of features for an applications audio system.

Below is a diagram showing 2 possible types of integration that can occur when using the ISACT engine component.



Sample Banks

The ISACT engine organizes sample data buffers into groups called Sample Banks. Sample Bank files are created with the IPS software. When loading a Sample Bank into the ISACT engine you must supply the zero based index of the Sample Bank slot you wish to load the Bank into. If there is already a Sample Bank loaded into the specified slot, it is replaced by the new Bank.

ISACT can manage up to 256 Sample Banks at one time. Each Sample Bank can contain any number of Samples. Each Sample within a Sample Bank can be identified by its zero based index. Samples are not necessarily consecutive; it is dependent on how the Sample Bank was made. Below is a diagram showing how Sample Banks with Samples are stored in ISACT.

Sample Bank Slot 000	mysamples.isb
	000 bang.wav
	001 <empty>
	002 step.wav
Sample Bank Slot 001	<empty>
Sample Bank Slot 002	moresamples.isb
	000 <empty>
	001 explosion.wav
Sample Bank Slot ...	<empty>
Sample Bank Slot 255	<empty>

These functions are used to manipulate Sample Banks.
LoadSampleBank
LoadStreamingSampleBank
UnloadSampleBank
GetSampleBankIndex
IsValidSampleBankIndex
GetSampleBankName

These functions are used to manipulate samples.
GetSampleHandle
IsValidSampleHandle
GetSampleName

The ISACT Production Studio enables the Sound Designer to give audio samples a destination target compression format, e.g ADPCM or Ogg Vorbis. The compression is done using a stand-alone tool called BankBuilder. In addition to performing sample compression, the BankBuilder strips ISACT project files (Sample Banks, Content Banks and SAC Files) of any information that is not needed by the run-time engine. See the section on BankBuilder for usage instructions.

Content Banks

At a higher level ISACT introduces new types of playable content. These new types of content offer many more features above just playing a basic sample. ISACT organizes these pieces of content into Content Banks. ISACT can manage any number of Content Banks. Content Banks are referenced by handles. When a Content Bank is loaded, ISACT returns a handle to the application to use for referencing the bank.

These higher level pieces of content reference Samples using Sample Bank and Sample indices. For the content in a Content Bank to produce the proper audio sound, the proper Sample Banks must also be loaded into the proper Sample Bank slots.

These functions are used to manipulate Content Banks.
LoadContentBank
UnloadContentBank
GetContentBankHandle
IsValidContentBankHandle
GetNextContentBank
GetContentBankName

These functions are used to manipulate pieces of content in a Content Bank.
GetContentHandle
IsValidContentHandle
GetContentMarker
GetContentName

The SAC File

A SAC file is a combonation of a Sample Bank and a Content Bank. There can only ever be one SAC file loaded at a time. Loading a new SAC file replaces the old one. A SAC file is a way for the Content Creator to make one file that includes everything for simple projects.

To access the SAC file Sample Bank the pre defined value ISACT_SAC_SAMPLE_BANK can be used. This value can be used with the following functions.

IsValidSampleBankIndex
GetSampleBankName
GetSampleHandle
GetSampleName

To access the SAC file Content Bank a NULL can be used in place of a Content Bank handle (IHANDLE). The following functions will alow a NULL to be used to acces the SAC file Content Bank.

IsValidContentBankHandle
GetContentBankName
GetContentHandle
GetNextContent
GetContentName
AttachPlayerRPC
TransitionAudioPlayer

A SAC file can also include references to other Sample Bank and Content Bank files to be loaded. When the SAC file is loaded the ISACT engine will also load these referenced files as well. This allows the Content Creator to create level files that can reuse assets. For instance there may be some Sample Banks that always need to be loaded for every level such as footsteps and weapons, and some that are level specific. Creating a SAC file is also a good way to ensure that the proper Sample Banks get loaded into the proper slots for use by the playable content.

These functions are used to manipulate a SAC file.
LoadFile

Samples

Samples can be attached to a Player via the AttachPlayerPrimaryContent or AttachPlayerTransitionContent functions. To get a handle to a sample to attach to a player use the GetSampleHandle function. Samples work like a piece of playable content in that they can be attached and played on players. The main difference in using samples instead of playable content is in how the handle to the sample is retrieved. Samples are identified either by their index within the sample bank or by their name. A sample is the most primitive type of audio that can be played using the ISACT system. All other types of playable content reference samples. For more information on how to play a sample see the tutorial on "Playing a Sample".

Content

There are three types of Content that are used with the ISACT run-time engine; Playable Content, Controller Content, and Script-Based Content. Playable Content can be attached to, and played on Audio Players. Controller Content is used to control how the Playable Content is played on those Audio Players. Script-Based Content is used to manipulate Playable and Controller Content based on variables controlled by the application.

Playable Content

The ISACT Engine can play the following types of Content on an Audio Player: -

- Sequences
- Sound Events
- Sound Queues
- Global Effects
- Sound Timelines
- Sound Randomizers

From the Player perspective all Playable Content looks the same. It makes no distinction between content types. It simply plays the content and applies all properties given to the player such as position and/or orientation to the content it is playing.

Playable Content can be attached to a Player via the AttachPlayerPrimaryContent or AttachPlayerTransitionContent functions.

For more information on playing a piece of content see the tutorial on "Playing a piece of Content".

Controller Content

The following types of Controller Content are used to affect the Content playing on an Audio Player: -

- Real-time Parameter Controllers (RPC)
- Transitions

RPCs can be attached to a player with the AttachPlayerRPC function. For more information on using a RPC see the tutorial on “Using a RPC on a Player”.

Transitions are user with the function TransitionAudioPlayer. For more information on using transitions see the tutorial on “Transitioning a Player”.

Script-Based Content

Sound Entities are theonly script-based Content type defined. They are high-level objects that can be used to control other pieces of content, such as Sound Events, and Sequences. Sound Entities are made up of 3 parts – Variables, Players, and Actions.

Entity Variables

There are two types of variables; local variables and global variables. Local variables are specific to a particular Entity, and can only be modified by communicating with that Entity. Global variables however, are shared across all the Entities contained in the same Content Bank, and can be modified by any Entity in the Content Bank. Both types of variable contain a number of states, including a default state. Changing the state of an Entity's local variable will never cause any changes to other Entities. Changing the state of a global variable, may cause multiple Entities to re-act.

Entity Players

Entity Players are identical to Audio Players, and are used by the Entity to play audio Content. An Entity can specify that a particular Real-time Parameter Controller is attached to a Player, and it can also mark any Players as the Master Player, which other players can be synchronized too.

Entity Actions

An Entity Action is essentially a script file that contains a number of conditions, which if met, will render a number of results. Conditions are based on the states of variables, e.g. if a particular variable is equal to a state, or is less-than a state, etc … If the conditions evaluate to true, then the results are rendered. Results include playing, stopping, and transitioning content, setting position, velocity, orientation, head-relative mode, and pitch on any of the Players contained in the Entity. An Action can also result in changing the state of any of the Entity's variables (which can result in further Actions be evaluated).

Entities are controlled by changing the states of variables. When a variable is set to a new state, the Entity runs through all the Action scripts that refer to the changed variable, and if the conditions are met, the results are rendered.

For more information on using an Entity see the tutorial on "Creating an Entity".

Header Files

The ISACT Production Studio can export a C header file containing information about the Content objects and Audio Samples contained in Project files (*.sac), Content Bank files (*.icb) and Sample Bank files (*.isb).

An IPS generated header file, can contain the following information:

- #define for the Project Filename
- Array of referenced Content Bank filenames
- Array of referenced Content Bank names
- Array of referenced Sample Bank filenames
- Array of referenced Sample Bank names
- #define's for the number of each type of Bank referenced
- Array of Content names and Sample names (including sample index) for the Project File
- Arrays of Content names for each Content Bank referenced
- Arrays of Sample names and indices for each Sample Bank referenced

Bank Builder

BankBuilder is a separate command-line application that is used to prepare ISACT assets for delivery. It has two main purposes; it is used to perform sample compression based on the settings made by the Sound Designer using ISACT Production Studio (IPS) and to strip ISACT files of any data that is only required by the IPS, rather than the ISACT runtime engine.

Currently supported compression formats are IMA 4bit ADPCM (64+1 samples per block), Ogg Vorbis (requires Ogg Vorbis DLLs to be in the BankBuilder directory), and Microsoft's WMA (requires WMSDK Runtime to be installed). Ogg Vorbis DLLs can be downloaded from www.xiph.org. WMSDK redistributatable can be downloaded from www.microsoft.com/windows/windowsmedia/default.aspx.Note that there are additional license requirements for applications wishing to distribute the Ogg Vorbis or WMA components, please refer to their websites mentioned above for more details.

The BankBuilder command line application takes the following optional parameters: -

-i<Input Directory>	: Directory to search for ISACT Files
-o<Output Directory>	: Directory to save the built ISACT Files
-r	: Recurse into sub-directories
-f<Filename>	: Filename (with or without wildcards)
-c<CompressionType>	: Force compression of all samples into format (see below)
-q<0.0 to 1.0>	: Compression Quality (1.0 is best). (Not relevant for ADPCM)
-h	: Display help

Where <CompressionType> can be IMA4ADPCM or OGGVORBIS or WMA.

Example 1

Build c:\input\myproject.sac and save the result in c:\output

BankBuilder -ic:\input\ -oc:\output -fmyproject.sac

Example 2

Build all the Sample Banks files in c:\samplebanks (and subdirectories) and save in c:\build

BankBuilder -ic:\samplebanks -oc:\build -r -f*.isb

Example 3

Convert all the samples in ISACT file types in c:\ISACTFiles (and subdirectories) into ADPCM and save in c:\build

BankBuilder -ic:\ISACTFiles -oc:\build -r -f*.* -cIMA4ADPCM

Currently Sample Banks that have been built cannot be loaded back into the IPS software. Built banks can only be used with the runtime engine.

Audio Players

The ISACT Audio Player is used to play Samples and Playable Content. It is an object that keeps track of an instance of playing audio and all of the properties associated with it such as the voices needed to play the content. Creating a player however does not use any audio voices. Voices are only allocated when the player needs to play the audio.

Before any audio can be played, a player must be created to play the audio. Once created audio content must then be attached to the player. After the content has been attached it can then be played. When a player is no longer needed it should be destroyed.

An audio player would generally be associated with an object or entity in a game such as an enemy creature or the game player character. In some cases it may be desirable to associate more than one audio player with a game object. Two players might be used for an enemy creature, one for footsteps sounds and one for speech. An audio player may also be associated with the different audio tracks in a game such as one for the main music track and one for the ambient sounds track.

To create a player, use the CreateAudioPlayer method. When a player is no longer needed perhaps because the object it is associated with is gone it should be destroyed by using the DestroyAudioPlayer method.

There are 2 methods for attaching content to a player, AttachPlayerPrimaryContent and AttachPlayerTransitionContent. For either type, if content is already attached it will be replaced.

There are 3 methods for playing content attached to the player, PlayAudioPlayer, TransitionAudioPlayer and RestartAudioPlayer.

In addition, there are 2 methods for attaching and controlling Real-time Parameter Controllers on a player, AttachPlayerRPC, and SetAudioPlayerRPCLevel.

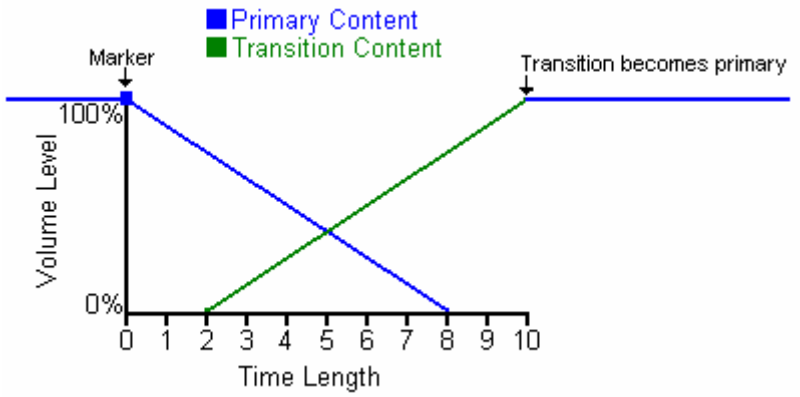
Transitioning

A single player can handle 2 pieces of content called the *Primary Content* and the *Transition Content*. This allows the player to transition from one piece of audio to another. A player can transition from the primary content to the transition content in a predefined manner. When a transition is complete the primary content is released and the transition content becomes the new primary content.

A transition is defined by the following properties.

- When it starts (this is in relation to the primary content).
- The total time length of the transition.
- The time offset within the transition that the primary content should end.
- The volume level percentage the primary content should end at.
- The time offset within the transition the transition content should start.
- The volume level percentage the transition content should start at.

Here is a graphical representation of a hypothetical 10 second long player transition.



This transition started when the primary content reached its next marker during playback. Over the first 8 seconds of the transition the primary contents volume was linearly interpolated from 100% to 0% and then stopped. The transition content was started 2 seconds into the transition with a starting volume level of 0%. Over the final 8 seconds the transition contents volume was linearly interpolated to 100% and then became the primary content.

To make a transition, use the TransitionAudioPlayer function. There are 3 versions of this function. The first 2 require the name of a pre created piece of Transition content. The third requires a STransition structure.

For more information on using transitions see the tutorial on "Transitioning a Player".

Real-Time Parameter Controller

A Real-Time Parameter Controller (or RPC) can be used to dynamically update properties of a Player based on a single input level. The RPC Content consists of a number of graphs that specify how a Player property, will be controlled based on the value of the RPC input level. RPC's can affect Volume, Pitch, Direct, Direct HF, and Send and Send HF levels to each FX Slot. The RPC input level; set using SetAudioPlayerRPCLevel, is a floating point value between 0 and 1.

An RPC is attached to any Player using AttachPlayerRPC and removed using ReleasePlayerRPC.

For more information on using a RPC see the tutorial on "Using a RPC on a Player".

Audio Groups

An Audio Group is simply a named selection of ISACT Content objects and Samples. The ISACT Production Studio (IPS) allows the user to create any number of Groups, and to assign any / all playable Content and Samples to any of those Groups. One piece of Content can only belong to one Group.

At load-time, as each piece of Content, or Audio Sample is loaded, if it has been assigned to a Group, then a Group of that name is created (if it doesn't exist already), and a link between the object and the group is stored. As Groups are created based on name, it is possible to use the same Group name in multiple Bank files, to group together audio stored in different Bank files.

The run-time engine supports a number of functions for dealing with audio on a per-Group basis. Specifically, the engine allows all content in a Group to be Paused (PauseAllAudio), Re-started (RestartAllAudio), or Stopped (StopAllAudio). If a Group includes any Sound Entities, then all the Audio Players contained in those Entitites will be paused, re-started, or stopped appropriately.

In addition to controlling playback of content in Groups, a group volume can be applied to all the content in a group using SetGroupGain. Group Gains are always combined with any existing gains on the Audio Player playing the Content, in the Content itself, or based on an attached Real-time Parameter Controller. Therefore, setting a Group Gain to 0 will result in silence, while a gain of 1 provides no additional attenuation. The SetGroupGain call includes an optional parameter for fade-time that can be used to ramp a Group volume to a new level over a given period of time.

Group methods use a Group Handle to identify a group, which is obtained using GetGroupHandle and passing in the name of a Group.

ISACT Game Audio Engine Tutorials

In the follwing sections there are 8 tutorials showing how to use many of the basic functions of the ISACT Gaming Engine.

Tutorial 1: Initializing the ISACT Engine

Before the ISACT Engine Interface can be created, a rendering device interface must be created for ISACT to use. The ISACT SDK comes with rendering components for each of the supported platforms. For each component an exported function called RetrieveISACTRenderInterface is exported to retrieve this interface. The ISACT engine component also exports a function called RetrieveISACTInterface to get the actual ISACT engine interface. Below is a code example of how to get the render interface and create the engine interface.

```
#include <isactri.h>
#include <isacteng.h>

ISACTRenderInterface *g_pISACTRenderInterface;
ISACTInterface *g_pISACTInterface;


// Place the following code in your initialization routine.

g_pISACTRenderInterface = RetrieveISACTRenderInterface();
if ( g_pISACTRenderInterface == NULL )
{
   // Handle error.
   return;
}

g_pISACTInterface = RetrieveISACTInterface(g_pISACTRenderInterface);
if ( g_pISACTInterface == NULL )
{
   // Handle error.
   g_pISACTRenderInterface->ReleaseInterface();
   return;
}

// Initialization successful.
```

Tutorial 2: Loading Files into the ISACT Engine

Once the ISACT engine has been initialized you will want to load in some data. ISACT can load 3 different types of files. Sample Bank (.isb), Content Bank (.icb), and Spatial Audio Composition (.sac) files.

If all your application will do is trigger simple samples (wave files) then you can load in Sample Bank files which are collections of samples packaged together. When loading a Sample Bank you need to supply the index of the bank to load into. If that bank index is already loaded, the new bank will replace the previous one. You load sample banks using the LoadSampleBank function. Below is a code example of how to load a Sample Bank file called test.isb into bank 0.

```
// This assumes the ISACT engine interface is already valid.

ISACTRETURN ir = g_pISACTInterface->LoadSampleBank(0, “test.isb”);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// File loaded successfully.
```

Content Bank files contain more complex sounds such as Sound Events and Sequences. You can load Content Bank files using the LoadContentBank function. Content Banks are stored as a linked list and are referred to by their handle. Content Banks reference samples in one or more Sample Banks. Loading a Content Bank without loading any Sample Banks will result in no audio ever being heard. Loading Content Banks individually requires loading the proper Sample Bank or Banks into the proper bank indices. Below is a code example showing how to load a Content Bank called test.icb.

```
// This assumes the ISACT engine interface is already valid.

IHANDLE g_iContentBankHandle;
ISACTRETURN ir = g_pISACTInterface->LoadContentBank(&g_iContentBankHandle, “test.icb”);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// File loaded successfully.
```

Spatial Audio Composition or SAC files are a combination of a Content Bank and a Sample Bank. SAC files can also contain references to other Content Bank and Sample Bank files that will also be loaded as a result of loading a SAC file. Loading a SAC file causes any and all previously loaded files to be cleared. SAC files are a good way of organizing different file sets of Sample Banks and Content Banks for say a single game level. You can load a SAC file using the LoadFile function. Below is a code example showing how to load a SAC file called test.sac. The file contains a reference to a Content Bank file called test.icb that will also be loaded. To use the Content Bank you will need to get the handle for that bank. This can be done using the GetContentBankHandle function.

```
// This assumes the ISACT engine interface is already valid.

ISACTRETURN ir = g_pISACTInterface->LoadFile( “test.sac”);
if ( ir != ISACT_OK )
{
    // Handle error.
    return;
}

IHANDLE g_iContentBankHandle;
ir = g_pISACTInterface->GetContentBankHandle(&g_iContentBankHandle , “test.icb”);
if ( ir != ISACT_OK )
{
    // Handle error.
    return;
}

// File loaded successfully.
```

Tutorial 3: Playing a Sample

The most basic audio that can be played using the ISACT engine is the sample. Samples come from preloaded Sample Banks or SAC file. After loading some samples into the engine you can get a handle to the sample by index or by name using the GetSampleHandle function. To play the sample you must then create a player to play it on using the CreateAudioPlayer function. Once a player is created you can attach the sample to the player using the AttachPlayerPrimaryContent function and play it using the PlayAudioPlayer function. Below is a code example showing how to get the sample handle by name, then create a player, attach and play it.

```
// This assumes the ISACT engine interface is already valid and a sac file with the sample
// test.wav in its sample bank has been loaded.

IHANDLE iSampleHandle;
ISACTRETURN ir = g_pISACTInterface->GetSampleHandle(&iSampleHandle,  “test.wav”,
                                                    ISACT_SAC_SAMPLE_BANK);
if ( ir != ISACT_OK )
{
   // Missing sample.
   return;
}

IHANDLE iPlayerHandle;
ir = g_pISACTInterface->CreateAudioPlayer(&iPlayerHandle);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

ir = g_pISACTInterface->AttachPlayerPrimaryContent(iPlayerHandle, iSampleHandle);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

ir = g_pISACTInterface->PlayAudioPlayer(iPlayerHandle, 0, NULL);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Sample playing successfully
```

Tutorial 4: Playing a piece of Content

This tutorial shows how to play a piece of playable content. Content comes from preloaded Content Banks or a SAC file. After loading some content into the engine you can get a handle to the content by index or by name using the GetContentHandle function. To play the content you must then create a player to play it on using the CreateAudioPlayer function. Once a player is created you can attach the content to the player using the AttachPlayerPrimaryContent function and play it using the PlayAudioPlayer function. Below is a code example showing how to get the content handle by name, then create a player, attach and play it.

```
// This assumes the ISACT engine interface is already valid and a sac file with the content
// footsteps in its content bankt has been loaded.

IHANDLE iContentHandle;
ISACTRETURN ir = g_pISACTInterface->GetContentHandle(&iContentHandle,  “footsteps”,
                                                                        NULL);
if ( ir != ISACT_OK )
{
   // Missing content.
   return;
}

IHANDLE iPlayerHandle;
ir = g_pISACTInterface->CreateAudioPlayer(&iPlayerHandle);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

ir = g_pISACTInterface->AttachPlayerPrimaryContent(iHandlePlayer, iContentHandle);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

ir = g_pISACTInterface->PlayAudioPlayert(iHandlePlayer, 0, NULL);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Content playing successfully
```

Tutorial 5: Transitioning a Player

Transitioning a player allows you to smoothly change the audio being played on the player. As an example, there could be a player associated with the engine of a car. When the engine is started it must transition form the starter sound to the engine idle sound. In stead of just abruptly changing the sound it would be better to do a quick crossfade. Below is a code example that shows how to do such a transition. In this example a STransition structure is used to describe the transition. Alternatively a pre made transition can be used and retrieved from a content bank by name. See the TransitionAudioPlayer function for more information.

```
// This assumes the ISACT engine interface is already valid and iCarPlayerHandle is a valid
// player handle. Also iStarterHandle and iIdelHandle are valid content handles.
// footsteps in its content bankt has been loaded.

// Attach the starter sound.
ISACTRETURN ir = g_pISACTInterface->AttachPlayerPrimaryContent(iCarPlayerHandle,
                                                               iStarterHandle);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Start playing the starter sound in looping mode.
ir = g_pISACTInterface->PlayAudioPlayert(iHandlePlayer, LOOP_INFINITE, NULL);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Sleep for 2 seconds.
sleep(2000);

// Attach the idle contentas the transition content
ISACTRETURN ir = g_pISACTInterface->AttachPlayerTransitionContent(iCarPlayerHandle,
                                                                  iIdleHandle);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Transition the player over 1 second to the looping idle sound.
STransition stFade = {START_IMMEDIATE, 1.0, 1.0, 0, 0, 1.0};
ISACTRETURN ir = g_pISACTInterface->TransitionAudioPlayer(iCarPlayerHandle,
                                                          &stFade,
                                                          LOOP_INFINITE,
                                                          NULL);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}
```

Tutorial 6: Using a RPC on a Player

A RPC is a controller that is created in IPS that can be used to alter parameters of the content playing on a player. An example of this could be the reving of an engine sound. It is common practice to alter the pitch of a looping engine idle sound to simulate the revving of the engine. A RPC can offer more than this as they can also control things like volume and effect sends. To use an RPC you must first attach it to the player. Below is a code example showing how to attach and use a RPC on a player.

```
// This assumes the ISACT engine interface is already valid and iCarPlayerHandle is a valid
// player handle that has a looping engine idle sound playing on it. It is also assumed that a SAC
file is loaded with a RPC called revcontrol in its content bank.

// Attach the starter sound.
ISACTRETURN ir = g_pISACTInterface->AttachPlayerRPC(iCarPlayerHandle,
                                                    “revcontrol”,
                                                    NULL,
                                                    0);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Adjust the rev level to half way.
ISACTRETURN ir = g_pISACTInterface->SetAudioPlayerRPCLevel(iCarPlayerHandle,
                                                           0.5f”);
```

Tutorial 7: Creating and using an Entity

This tutorial shows how to create an instance of a Sound Entity, activate it, and control actions by setting a variable to a particular state. An Entity instance is created from a Sound Entity stored in a Content Bank or SAC file. The CreateEntity method takes a name of a Sound Entity and a handle to a Content Bank containing that Sound Entity. After an Entity has been created, it needs to be activated using ActivateEntity before the state machine responds to variable state changes. Once activated, whenever a variable is set to a new state, the Entity script is run, and its results are rendered. To stop processing on an Entity, simply de-activate or destroy it.

```
// This assumes the ISACT engine interface is already valid and a sac file with the 'Music'
// Entity in its content bank has been loaded.

IHANDLE iEntityHandle;
ISACTRETURN ir = g_pISACTInterface->CreateEntity(&iEntityHandle, "Music", NULL);
if ( ir != ISACT_OK )
{
   // Missing Entity.
   return;
}

// Entities are in-active by default, so ActiveEntity now
ir = g_pISACTInterface->ActivateEntity(iEntityHandle, true);
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Set an Entity variable called 'Music' to a state called 'Start'
ir = g_pISACTInterface->SetEntityLocalVarState(iEntityHandle, "Music", "Start");
if ( ir != ISACT_OK )
{
   // Handle error.
   return;
}

// Entity created and activated with a variable set to a particular state

// To stop Entity, de-activate it
g_pISACTInterface->ActivateEntity(iEntityHandle, false);

// To clean-up destroy the Entity
g_pISACTInterface->DestroyEntity(iEntityHandle);
```

Tutorial 8: Using Groups

Every playable Content object in ISACT can be assigned to a Group. The Groups can be used to provide a quick and easy way to control volume levels of related sounds; e.g by assigning all of the voice samples used by an application to a “VoiceOver” Group, the application can use one function call to adjust the volume levels of all the voice samples. To use Groups, you need to get hold of a Group Handle, which is done using GetGroupHandle and passing in the name of the Group. This handle is used in any function calls that use Groups.

```
// This assumes the ISACT engine interface is already valid and a file has been loaded
// containing a Group called “VoiceOver”

ISACTRETURN ir;
IHANDLE iVoiceGroup;

ir = g_pISACTInterface->GetGroupHandle(&iVoiceGroup, "VoiceOver");
if ( ir != ISACT_OK )
{
   // Missing Group
   return;
}

// Set Group Volume to half
ir = g_pISACTInterface->SetGroupGain(iVoiceGroup, 0.5f);

// Pause all audio in the VoiceOver Group
ir = g_pISACTInterface->PauseAllAudio(iVoiceGroup);

// Resume all audio in the VoiceOver Group
ir = g_pISACTInterface->RestartAllAudio(iVoiceGroup);
```

Tutorial 9: Releasing the ISACT Engine

Before your application exists you must release the ISACT Engine to be sure all assets are freed. Below is a code example showing how to release the engine.

```
// This assumes the ISACT engine interface and the ISACTRendering interface are valid.

// Frist release the engine.
if (g_pISACTInterface )
{
    g_pISACTInterface->ReleaseInterface();
    g_pISACTInterface = NULL;
}

// Next release the rendering interface.
if (g_pISACTRenderInterface )
{
    g_pISACTRenderInterface->ReleaseInterface();
    g_pISACTRenderInterface = NULL;
}

// Engine released.
```

ISACT Game Audio Engine Interface

Audio Interface	RetrieveISACTRenderInterface
	GetNumISACTDevices
	GetISACTDeviceCaps
	OpenISACTDevice
	CloseISACTDevice
ISACT Interface	RetrieveISACTInterface
	ReleaseInterface
File Management	LoadFile
	ReleaseAllResources
	LoadContentBank
	UnloadContentBank
	SetContentBankSampleBankOffset
	LoadSampleBank
	LoadStreamingSampleBank
	LoadSampleBankEx
	UnloadSampleBank
Content Management	GetContentBankHandle
	IsValidContentBankHandle
	GetContentHandle
	IsValidContentHandle
	GetContentMarker
	GetContentVoiceCount
	GetContentMaxGain
	GetContentInfo
Content Enumeration	GetNextContentBank
	GetContentBankName
	GetNextContent
	GetContentName
Sample Management	GetSampleBankIndex
	IsValidSampleBankIndex
	GetSampleHandle
	IsValidSampleHandle
	CacheContentSamples
	CacheEntitySamples
	CacheStreamingSample
	UpdateStreamingSampleBanks
Sample Enumeration	GetSampleBankName
	GetSampleName
Audio	GetAvailableVoiceCount
	StopAllAudio
	PauseAllAudio
	RestartAllAudio
	SetHeadRoomGain
	GetHeadRoomGain
Distance	SetDistanceFactor
	GetDistanceFactor
Listener	SetListenerGain

	SetListenerPosition
	SetListenerOrientation (Vector based)
	SetListenerOrientation (Euler based)
	GetListenerPosition
	GetListenerOrientation (Vector based)
	GetListenerOrientation (Euler based)
EAX	SetEAXProperty
	GetEAXProperty
Player Handles	CreateAudioPlayer
	DestroyAudioPlayer
	IsValidPlayerHandle
Player Content	AttachPlayerPrimaryContent
	AttachPlayerTransitionContent
	AttachPlayerRPC
	ReleasePlayerPrimaryContent
	ReleasePlayerTransitionContent
	ReleasePlayerRPC
Synchronization	SetMasterAudioPlayer
	GetMasterAudioPlayer
Player Status	PlayAudioPlayer
	TransitionAudioPlayer
	PauseAudioPlayer
	RestartAudioPlayer
	StopAudioPlayer
	GetAudioPlayerStatus
	GetAudioPlayerMetricTime
	GetAudioPlayerRealTime
Player Properties	SetAudioPlayerMode
	SetAudioPlayerPosition
	SetAudioPlayerOrientation (Vector based)
	SetAudioPlayerOrientation (Euler based)
	SetAudioPlayerVelocity
	SetAudioPlayerRPCLevel
	GetAudioPlayerMode
	GetAudioPlayerPosition
	GetAudioPlayerOrientation (Vector based)
	GetAudioPlayerOrientation (Euler based)
	GetAudioPlayerVelocity
	GetAudioPlayerRPCLevel
	GetAudioPlayerMaxGain
Entity Handles	CreateEntity
	DestroyEntity
	IsValidEntityHandle
Entity manipulation	ActivateEntity
	IsEntityPlaying
	ResetEntityLocalVars
	ResetEntityGlobalVars
	SuspendEntity
	ProcessEntity
	SuspendAllEntities
	ProcessAllEntities
	SetEntityLocalVarState (String based)
	SetEntityGlobalVarState (String based)

	SetEntityLocalVarState (Index based)
	SetEntityGlobalVarState (Index based)
Entity Properties	SetEntityMasterRPCLevel
	SetEntityRPCLevel
	SetEntityMode
	SetEntityPosition
	SetEntityOrientation (Vector based)
	SetEntityOrientation (Euler based)
	SetEntityVelocity
	SetEntityPitch
	SetEntityGain
	GetEntityLocalVarState (String based)
	GetEntityGlobalVarState (String based)
	GetEntityLocalVarState (Index based)
	GetEntityGlobalVarState (Index based)
	GetEntityMasterRPCLevel
	GetEntityRPCLevel
	GetEntityMode
	GetEntityPosition
	GetEntityOrientation (Vector based)
	GetEntityOrientation (Euler based)
	GetEntityVelocity
	GetEntityPitch
	GetEntityGain
	GetEntityMaxGain
Entity Query	GetNumLocalVars
	GetNumGlobalVars
	GetNumLocalVarStates
	GetNumGlobalVarStates
	GetLocalVarName
	GetGlobalVarName
	GetLocalVarStateName
	GetGlobalVarStateName
Groups	GetGroupHandle
	IsValidGroupHandle
	SetGroupGain
	GetGroupGain

Initialization and Release Functions

Audio Interface	RetrieveISACTRenderInterface
	GetNumISACTDevices
	GetISACTDeviceCaps
	OpenISACTDevice
	CloseISACTDevice
ISACT Interface	RetrieveISACTInterface
	ReleaseInterface

RetrieveISACTRenderInterface

The **RetrieveISACTRenderInterface** function is exported from AudioDrv.dll and is used to obtain the ISACTRenderingInterface compatible interface. Applications that wish to use their own custom ISACTRenderingInterface compatible audio engine will use their own function call for retrieving the interface.

```
ISACTRenderInterface* RetrieveISACTRenderInterface(
        void *pUserData = 0,
        long lVoiceCount = ISACT_MAX_VOICES
);
```

Parameters

pUserData
A pointer to an Open AL Context, or NULL for the Rendering Interface to create its own Open AL Device and Context.

lVoiceCount
Maximum number of voices to create.

Return Values

Pointer to an ISACTRenderInterface object.

Remarks

The number of voices created may be less than *lVoiceCount* if the rendering platform does not support that many voices.

An audio engine that derives from the ISACTRenderInterface class, and implements all the methods, must be passed to the RetrieveISACTInterface function call used to initialize the ISACT run-time engine.

See Also

RetrieveISACTInterface

GetNumISACTDevices

The **GetNumISACTDevices** function is exported from AudioDrv.dll and can be used to determine the number of Open AL Devices on the system that can be used by the ISACTRenderer. Applications that wish to use their own custom ISACTRenderingInterface compatible audio engine are expected to use their own enumeration system to select an output device.

```
unsigned long GetNumISACTDevices();
```

Parameters

None

Return Values

The number of Open AL devices on the system.

Remarks

This helper function will enumerate all of the available Open AL Devices on the system and collect information about the capabilities of each device. Device information can be retrieved using GetISACTDeviceCaps. Once an appropriate Device has been selected, it can be opened with OpenISACTDevice, which returns an Open AL Context pointer that can be passed to RetrieveISACTRenderInterface. When an application terminates, it should release the ISACT Interface, then the ISACT Rendering Interface, and finally close the AL device using CloseISACTDevice.

See Also

GetISACTDeviceCaps, OpenISACTDevice, CloseISACTDevice, RetrieveISACTRenderInterface

GetISACTDeviceCaps

The **GetISACTDeviceCaps** function is exported from AudioDrv.dll and can be used to retrieve the capabilities of one of the devices that were enumerated using GetNumISACTDevices. Applications that wish to use their own custom ISACTRenderingInterface compatible audio engine are expected to use their own enumeration system to select an output device.

```
bool GetISACTDeviceCaps(
        unsigned long ulDeviceNum,
        ISACTDeviceCaps *pDeviceCaps
);
```

Parameters

ulDeviceNum
Index of Device to query.

pDeviceCaps
Address of an ISACTDeviceCaps structure to be filled with the capabilities of the AL Audio Device.

Return Values

True if pDeviceCaps is a valid pointer, and ulDeviceNum is a valid index, false otherwise.

Remarks

This helper function returns the capabilities of the given Open AL Device. An application must call GetNumISACTDevices before calling this function.

See Also

GetNumISACTDevices, OpenISACTDevice, CloseISACTDevice, RetrieveISACTRenderInterface

OpenISACTDevice

The **OpenISACTDevice** function is exported from AudioDrv.dll and can be used to open a previously enumerated AL device and prepare if for use with the ISACT Renderer. Applications that wish to use their own custom ISACTRenderingInterface compatible audio engine are expected to use their own enumeration system to select an output device.

```
void *OpenISACTDevice(
        ISACTDeviceCaps *pDeviceCaps
);
```

Parameters

pDeviceCaps
Address of an ISACTDeviceCaps structure that was filled by GetISACTDeviceCaps.

Return Values

The void pointer returned is a pointer to an Open AL Context, which can be passed to RetrieveISACTInterface.

Remarks

This helper function opens the device referenced in the ISACTDeviceCaps structure that was returned by GetISACTDeviceCaps. If this call is successful, it will release the memory used to store information about each ISACT Device, meaning that calls to GetISACTDeviceCaps will no longer be valid (without re-calling GetNumISACTDevices).

See Also

GetNumISACTDevices, GetISACTDeviceCaps, CloseISACTDevice, RetrieveISACTRenderInterface

CloseISACTDevice

The **CloseISACTDevice** function is exported from AudioDrv.dll and can be used to close a previously opened AL device. Applications that wish to use their own custom ISACTRenderingInterface compatible audio engine are expected to use their own enumeration system to select an output device.

```
void CloseISACTDevice(
        void *pData
);
```

Parameters

pData
The void pointer returned from OpenISACTDevice.

Return Values

None.

Remarks

This helper function closes the AL device that was opened using OpenISACTDevice. The application should release the ISACTRendering interface before attempting to close the AL Device.

See Also

GetNumISACTDevices, GetISACTDeviceCaps, OpenISACTDevice, RetrieveISACTRenderInterface

RetrieveISACTInterface

The **RetrieveISACTInterface** function is exported from ISACTWin.dll and is used to obtain the main ISACT interface, which contains all the methods for handling Files, Content, Samples, Players and Entities.

```
ISACTInterface* RetrieveISACTInterface(
        ISACTRenderInterface *pRenderInterface
);
```

Parameters

pRenderInterface
Pointer to an ISACTRenderInterface compatible audio engine.

Return Values

Pointer to an ISACTInterface object.

Remarks

The ISACTRenderInterface pointer passed to this method can be an application defined audio engine (deriving from ISACTRenderInterface), or the audio engine shipping with ISACT (in AudioDrv.dll), obtained using RetrieveISACTRenderInterface.

See Also

RetrieveISACTRenderInterface

ISACTInterface::ReleaseInterface

The **ISACTInterface::ReleaseInterface** method is used to shutdown the ISACT engine and release all associated memory. The ISACTRenderInterface (passed to the ISACTInterface at creation time) is not released, so the application should destroy this object itself.

```
void ReleaseInterface();
```

Parameters

None

Return Values

None

Remarks

The ISACTInterface is destroyed by this call, so an application must re-initialize ISACT using RetrieveISACTInterface before making any ISACT calls.

See Also

RetrieveISACTInterface

File and Content Management Functions

File Management	LoadFile
	ReleaseAllResources
	FileOverride
	LoadContentBank
	UnloadContentBank
	IsContentBankInUse
	SetContentBankSampleBankOffset
	LoadSampleBank
	LoadStreamingSampleBank
	LoadSampleBankEx
	UnloadSampleBank
Content Management	GetContentBankHandle
	IsValidContentBankHandle
	GetContentHandle
	IsValidContentHandle
	GetContentMarker
	GetContentVoiceCount
	GetContentMaxGain
	GetContentInfo
Content Enumeration	GetNextContentBank
	GetContentBankName
	GetNextContent
	GetContentName
Sample Management	GetSampleBankIndex
	IsValidSampleBankIndex
	GetSampleHandle
	IsValidSampleHandle
	CacheContentSamples
	CacheEntitySamples
	CacheStreamingSample
	UpdateStreamingSampleBanks
Sample Enumeration	GetSampleBankName
	GetSampleName

ISACTInterface::LoadFile

The **ISACTInterface::LoadFile** method is used to load .sac files previously created with the ISACT Production Studio. All of the Content Banks and Sample Banks referenced in the .sac file will be loaded.

```
ISACTRETURN LoadFile(
        const char *szFileName
);
```

```
ISACTRETURN LoadFile(
        const unsigned short *szFileName
);
```

Parameters

szFileName
NULL terminated filename, including path, to the .sac file to load.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_FILE_OPEN_ERROR	File not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized
ISACT_CONTENTBANK_NOT_FOUND	Referenced Content Bank not found
ISACT_SAMPLEBANK_NOT_FOUND	Referenced Sample Bank not found

Remarks

Only one .sac file may be loaded at once, so calling LoadFile will stop all the Audio Players and remove any attached content, destroy any existing Entities, and unload all existing Content and Sample Banks.

If one or more of the referenced Content or Sample Banks cannot be found, then the method fails with an appropriate error code, and no audio content is loaded. Content and Sample Bank references are stored in a .sac file as relative file paths from the .sac file itself, so it is important to maintain the same relative directory structure as was used when the files were created in the ISACT Production Studio (IPS).

An application can override ISACT's file loading operations by calling FileOverride.

See Also

ReleaseAllResources

ISACTInterface::ReleaseAllResources

The **ISACTInterface::ReleaseAllResources** method is used to remove all loaded content (either from LoadFile, or from individual calls to LoadSampleBank and LoadContentBank).

```
ISACTRETURN ReleaseAllResources();
```

Parameters

None

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

This method will stop all the Audio Players and remove any attached content, destroy any existing Entities, and unload all existing Content and Sample Banks. All Content, Content Bank, Sample, Sample Bank and Entity handles are invalidated.

See Also

LoadFile, LoadSampleBank, LoadContentBank

ISACTInterface::FileOverride

The **ISACTInterface::FileOverride** method is used to replace ISACT's file reading functions with user-supplied functions.

```
ISACTRETURN FileOverride(
        SStreamFunctions *pStreamFunctions
);
```

Parameters

pStreamFunctions
Pointer to an SStreamFunctions structure containing the addresses of the file operation functions.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

If an application calls this function, all subsequent file loading operations will be routed to the user supplied functions. This enables an application to embed ISACT file data inside another file or in a compressed file format.

See Also

LoadFile, LoadSampleBank, LoadContentBank

ISACTInterface::LoadContentBank

The **ISACTInterface::LoadContentBank** method is used to load individual Content Banks files (*.icb).

```
ISACTRETURN LoadContentBank(
        IHANDLE *piHandleContentBank,
        const char *szFileName
);
```

```
ISACTRETURN LoadContentBank(
        IHANDLE *piHandleContentBank,
        const unsigned short *szFileName
);
```

Parameters

piHandleContentBank
Address of an IHANDLE that will receive the Content Bank's handle if loaded successfully.

szFileName
NULL terminated filename, including path, to the Content Bank file to load.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_MEMORY_ERROR	Out of memory
ISACT_FILE_OPEN_ERROR	File not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

After loading a Content Bank file, handles to the Content stored in the bank can be retrieved using GetContentHandle.

An application can override ISACT's file loading operations by calling FileOverride.

See Also

UnloadContentBank

ISACTInterface::UnloadContentBank

The **ISACTInterface::UnloadContentBank** method is used to unload a Content Bank previously loaded using LoadFile or LoadContentBank.

```
ISACTRETURN UnloadContentBank(
        IHANDLE iHandleContentBank
);
```

Parameters

iHandleContentBank
Handle of the Content Bank to unload.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_IN_USE	One or more pieces of Content are being used
ISACT_BAD_HANDLE	Bad or invalid handle

Remarks

If one or more pieces of Content in the Bank are being used, then the Content Bank cannot be unloaded. Content should be released from Audio Players, and Entities should be destroyed before unloading the bank.

See Also

LoadFile, LoadContentBank, IsContentBankInUse

ISACTInterface::IsContentBankInUse

The **ISACTInterface::IsContentBankInUse** method is used to determine if a Content Bank can be safely unloaded.

```
ISACTRETURN IsContentBankInUse(
        IHANDLE iHandleContentBank,
        unsigned long *pulReferenceCount
);
```

Parameters

iHandleContentBank
Handle of the Content Bank to check.

pulReferenceCount
Address of a variable to receive the number of objects still in use inside the Content Bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_BAD_HANDLE	Bad or invalid handle

Remarks

If one or more pieces of Content in the Bank are being used, then the Content Bank cannot be unloaded. Content should be released from Audio Players, and Entities should be destroyed before unloading the bank. This method can be used to query ISACT for the number of objects inside the Bank that are still in use.

See Also

LoadContentBank, UnloadContentBank

ISACTInterface::SetContentBankSampleBankOffset

The **ISACTInterface::SetContentBankSampleBankOffset** method is used to set the sample bank starting index used by content items in the bank.

```
ISACTRETURN SetContentBankSampleBankOffset(
        IHANDLE iHandleContentBank,
        unsigned long ulSampleBankOffset
);
```

Parameters

iHandleContentBank
Handle of the Content Bank to set the offset for.

ulSampleBankOffset
Offset value to be used when indexing Sample Banks.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Bad or invalid handle

Remarks

The offset value is added to each Sample Bank index used by each piece of content in the Content Bank. In this way a Content Bank that was created to use Sample Banks 1 and 2 can be made to use 3 and 4 instead by setting an offset value of 2. If the result of adding the original Sample Bank index and the offset together exceeds the maximum bank index value it is clamped.

To force a content bank to always use a specific sample bank index to retrieve all of its samples from, you can use the FORCE_SAMPLE_BANK macro on the ulSampleBankOffset parameter when using this function. For more information see FORCE_SAMPLE_BANK.

See Also

Sample Banks, Content Banks

ISACTInterface::LoadSampleBank

The **ISACTInterface::LoadSampleBank** method is used to load individual Sample Bank files (*.isb) into a specified Bank Index.

```
ISACTRETURN LoadSampleBank(
        unsigned long ulBankIndex,
        const char *szFileName,
        bool bEnableStreaming = true
);
```

```
ISACTRETURN LoadSampleBank(
        unsigned long ulBankIndex,
        const unsigned short *szFileName,
        bool bEnableStreaming = true
);
```

Parameters

ulBankIndex
The Bank index to load the Sample Bank into.

szFileName
NULL terminated filename, including path, of the Sample Bank file to load.

bEnableStreaming (default = true)
Enable / Disable streaming for Sample Banks marked as Streaming

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_MEMORY_ERROR	Out of memory
ISACT_FILE_OPEN_ERROR	File not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

If a Sample Bank already exists at the specified Bank Index, then the new Sample Bank overwrites the original Bank.

If a SampleBank includes Streaming Information, the run-time engine will stream all the samples from disc. The application can override this behaviour and disable streaming by setting the bEnableStreaming parameter to 'false'. If Streaming is enabled (the default) then the run-time engine will use the Streaming settings from the SampleBank file. If the SampleBank includes run-time decoding information, the run-time engine will use this information to determine whether to decode compressed audio at load-time or at run-time (play-time).

An application can override ISACT's file loading operations by calling FileOverride.

See Also

LoadSampleBankEx, LoadStreamingSampleBank, UnloadSampleBank

ISACTInterface::LoadStreamingSampleBank

The **ISACTInterface::LoadStreamingSampleBank** method is used to load individual Sample Bank files (*.isb) into a specified Bank Index as a streaming Sample Bank.

```
ISACTRETURN LoadStreamingSampleBank(
        unsigned long ulBankIndex,
        const char *szFileName,
        unsigned long ulPacketTimeLength,
        long lPreLoad = FORCE_SAMPLE_PRELOAD_FILE
);
```

```
ISACTRETURN LoadStreamingSampleBank(
        unsigned long ulBankIndex,
        const unsigned short *szFileName,
        unsigned long ulPacketTimeLength,
        long lPreLoad = FORCE_SAMPLE_PRELOAD_FILE
);
```

Parameters

ulBankIndex
The Bank index to load the Sample Bank into.

szFileName
NULL terminated filename, including path, of the Sample Bank file to load.

ulPacketTimeLength
The time length of a single stream packet in milliseconds (> 0 ms).

lPreLoad (default = FORCE_SAMPLE_PRELOAD_FILE)
Force Sample pre-loading on, off, or read the Sample's pre-load setting from the Sample Bank file.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_MEMORY_ERROR	Out of memory
ISACT_FILE_OPEN_ERROR	File not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

A streaming Sample Bank will stream all of its individual samples from disk instead of loading them into memory. If a Sample Bank already exists at the specified Bank Index, then the new Sample Bank overwrites the original Bank.

The application can use the lPreLoad parameter to choose to use the Sample's pre-load setting from the SampleBank file (FORCE_SAMPLE_PRELOAD_FILE), or force pre-loading on (FORCE_SAMPLE_PRELOAD_ON) or off (FORCE_SAMPLE_PRELOAD_OFF).

See Also

LoadSampleBankEx, LoadSampleBank, UnloadSampleBank, UpdateStreamingSampleBanks, CacheStreamingSample

ISACTInterface::LoadSampleBankEx

The **ISACTInterface::LoadSampleBankEx** method is used to load individual Sample Bank files (*.isb) into a specified Bank Index and configure streaming and run-time decode settings.

```
ISACTRETURN LoadSampleBankEx(
        unsigned long ulBankIndex,
        const char *szFileName,
        unsigned long ulFilePacketTimeLength = ISACT_ENABLE,
        long lPreLoad = FORCE_SAMPLE_PRELOAD_FILE,
        unsigned long ulRunTimeDecodePacketLength = ISACT_ENABLE
);
```

```
ISACTRETURN LoadSampleBankEx(
        unsigned long ulBankIndex,
        const unsigned short *szFileName,
        unsigned long ul ulFilePacketTimeLength = ISACT_ENABLE,
        long lPreLoad = FORCE_SAMPLE_PRELOAD_FILE,
        unsigned long ulRunTimeDecodePacketLength = ISACT_ENABLE
);
```

Parameters

ulBankIndex
The Bank index to load the Sample Bank into.

szFileName
NULL terminated filename, including path, of the Sample Bank file to load.

ulFilePacketTimeLength
The duration of a single file stream packet in milliseconds or ISACT_ENABLE to use the setting from the file, or ISACT_DISABLE to disable file streaming.

lPreLoad (default = FORCE_SAMPLE_PRELOAD_FILE)
Force Sample pre-loading on, off, or read the Sample's pre-load setting from the Sample Bank file.

ulRunTimeDecodePacketLength
The duration of a single decode stream packet in milliseconds or ISACT_ENABLE to use the setting from the file, or ISACT_DISABLE to disable run-time decoding.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_MEMORY_ERROR	Out of memory
ISACT_FILE_OPEN_ERROR	File not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

If a Sample Bank already exists at the specified Bank Index, then the new Sample Bank overwrites the original Bank.

The LoadSampleBankEx method gives the application complete control over file-streaming and run-time decoding. The default parameter values will tell the run-time engine to use the settings stored in the Bank File.

A streaming Sample Bank will stream all of its individual samples from disk instead of loading them into memory.

The application can use the lPreLoad parameter to choose to use the Sample's pre-load setting from the SampleBank file (FORCE_SAMPLE_PRELOAD_FILE), or force pre-loading on (FORCE_SAMPLE_PRELOAD_ON) or off (FORCE_SAMPLE_PRELOAD_OFF).

A Bank with run-time decoding enabled will no longer decode compressed samples at load-time. In the case of static Sample Banks, the compressed audio will be loaded and stored in RAM. In the case of streaming Sample Banks, this setting is irrelevant, unless the application chooses to cache the sample (see CacheStreamingSample).

See Also

LoadSampleBank, LoadStreamingSampleBank, UnloadSampleBank, UpdateStreamingSampleBanks, CacheStreamingSample

ISACTInterface::UnloadSampleBank

The **ISACTInterface::UnloadSampleBank** method is used to unload a Sample Bank that has been previously loaded using LoadFile or LoadSampleBank.

```
ISACTRETURN UnloadSampleBank(
        unsigned long ulBankIndex
);
```

Parameters

ulBankIndex
Sample Bank Index to unload.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

Content that references a Sample from this Bank will no longer play correctly.

See Also

LoadFile, LoadSampleBank

ISACTInterface::GetContentBankHandle

The **ISACTInterface::GetContentBankHandle** method is used to obtain a handle to a Content Bank that has been previously loaded using LoadFile or LoadContentBank.

```
ISACTRETURN GetContentBankHandle(
        IHANDLE *piHandleContentBank,
        const char *szContentBankName
);
```

```
ISACTRETURN GetContentBankHandle(
        IHANDLE *piHandleContentBank,
        const unsigned short *szContentBankName
);
```

Parameters

piHandleContentBank
Address of an IHANDLE that will receive the handle of the Content Bank.

szContentBankName
Name of the requested Content Bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	Object was not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The Content Bank handle is used to obtain handles to individual pieces of Content in the bank.

See Also

LoadFile, LoadSampleBank, IsValidContentBankHandle

ISACTInterface::IsValidContentBankHandle

The **ISACTInterface::IsValidContentBankHandle** method is used to determine if a particular handle is a valid Content Bank handle returned from GetContentBankHandle.

```
ISACTRETURN IsValidContentBankHandle(
        IHANDLE iHandleContentBank
);
```

Parameters

iHandleContentBank
Handle to validate.

Return Values

Identifier	Description
ISACT_OK	Content Bank handle is valid
ISACT_BAD_HANDLE	Content Bank handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

None.

See Also

LoadContentBank, GetContentBankHandle

ISACTInterface::GetContentHandle

The **ISACTInterface::GetContentHandle** method is used to obtain a handle to a piece of Content from a particular Content Bank.

```
ISACTRETURN GetContentHandle(
        IHANDLE *piHandle,
        const char *szContentName,
        IHANDLE iHandleContentBank
);
```

```
ISACTRETURN GetContentHandle(
        IHANDLE *piHandle,
        const unsigned short*szContentName,
        IHANDLE iHandleContentBank
);
```

Parameters

piHandle
Address of an IHANDLE that will receive the handle of the Content.

szContentName
Name of the requested Content.

iHandleContentBank
Handle of the Content Bank containing the Content, or the predefined value ISACT_SAC_CONTENT_BANK to access the SAC file content bank, or the predefined value ISACT_SEARCH_CONTENT_BANKS to search all of the currently loaded content banks for the content name.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_NOT_FOUND	Content object was not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

Content handles are used to attach Content to Audio Players, and to create Sound Entities. This function only returns handles to playable content. When using ISACT_SEARCH_CONTENT_BANKS the first found occurance of the content name is returned.

See Also

GetContentBankHandle, IsValidContentBankHandle, IsValidContentHandle

ISACTInterface::IsValidContentHandle

The **ISACTInterface::IsValidContentHandle** method is used to determine if a particular handle is a valid Content handle returned from GetContentHandle.

```
ISACTRETURN IsValidContentHandle(
        IHANDLE iHandleContent
);
```

Parameters

iHandleContent
Handle to validate.

Return Values

Identifier	Description
ISACT_OK	Content handle is valid
ISACT_BAD_HANDLE	Content handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

None.

See Also

GetContentHandle

ISACTInterface::GetContentMarker

The **ISACTInterface::GetContentMarker** method is used to obtain a handle to a Marker in the given Content.

```
ISACTRETURN GetContentMarker(
        IHANDLE *piHandle,
        char *szMarkerName,
        IHANDLE iHandleContent
);
```

```
ISACTRETURN GetContentMarker(
        IHANDLE *piHandle,
        unsigned short *szMarkerName,
        IHANDLE iHandleContent
);
```

Parameters

piHandle
Address of an IHANDLE that will receive the handle of the Marker.

szMarkerName
Name of the requested Marker.

iHandleContent
Handle of the Content containing the Marker.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Bad Content Handle
ISACT_NOT_FOUND	Marker not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

Not all Content can contain Markers.

See Also

GetContentHandle

ISACTInterface::GetContentVoiceCount

The **ISACTInterface::GetContentVoiceCount** method is used to determine how many voices the given Content requires to use when played.

```
ISACTRETURN GetContentVoiceCount(
        IHANDLE iHandleContent,
        long *plVoiceCount
);
```

Parameters

iHandleContent
Handle of the Content to query.

plVoiceCount
Address of a long variable that will receive the voice count of the Content.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Handle was not a valid Content Handle
ISACT_BAD_VALUE	One or more parameters are invalid or out-of-range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

Applications can use this and the GetAvailableVoiceCount method to determine if there are enough available voices to play the given Content.

See Also

GetAvailableVoiceCount, GetContentMaxGain, GetContentInfo

ISACTInterface::GetContentMaxGain

The **ISACTInterface::GetContentMaxGain** method is used to retrieve the loudest gain that this Content could have at a given 3D position.

```
ISACTRETURN GetContentMaxGain(
        IHANDLE iHandle,
        ISACTVector *pvPosition,
        unsigned long ulHeadRelativeMode,
        float *pflGain
);
```

Parameters

iHandle
Valid Content handle.

pvPosition
Address of an ISACTVector containing the position for the Content that will be used to calculate the attenuation over distance.

ulHeadRelativeMode
If true (set to 1) then the position pointed to by pvPosition will be treated as relative to the Listener's position, rather than an absolute co-ordinate.

pflGain
Address of a floating point variable that receives the Content's Max gain level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Content Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The MaxGain calculation takes into account distance attenuation (if any), and content gain (content level and sub-mix level). It does not include gain adjustments that are part of the content itself (e.g volume randomization in a Sound Event, or gain levels in a Sequence).

Note – to calculate the maximum gain of a piece of Content, ignoring 3D attenuation, pass in a vector of {0, 0, 0} and set ulHeadRelativeMode equal to 1.

See Also

GetEntityMaxGain, GetAudioPlayerMaxGain, GetContentVoiceCount, GetContentInfo

ISACTInterface::GetContentInfo

The **ISACTInterface::GetContentInfo** method is used to query the content object for information about the duration of playback. Many of the ISACT content objects feature random parameters and / or random selection of samples – so it is not always possible to compute exact playback duration; instead this method returns the shortest possible playback time and the longest possible playback time.

```
ISACTRETURN GetContentInfo(
        IHANDLE iHandle,
        SContentInfo *pContentInfo
);
```

Parameters

iHandle
Valid Content handle.

pContentInfo
Address of an SContentInfo structure to be filled in with Content object information.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Content Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Global Effect objects will always return minimum and maximum durations of 0 ms since their playback is completed instantanously. Sound Event objects that do no have a total of 100% chance of playing a track will always return a minimum duration of 0 ms. Sound Queues with a looped section will return a maximum duration of 0xFFFFFFFF ms because they will play forever.

See Also

GetContentMaxGain, GetContentVoiceCount, SContentInfo

ISACTInterface::GetNextContentBank

The **ISACTInterface::GetNextContentBank** method is used to enumerate all the Content Banks that have been loaded by LoadFile or LoadContentBank.

```
ISACTRETURN GetNextContentBank(
        IHANDLE *piHandleContentBank,
        IHANDLE iHandleContentBank
);
```

Parameters

piHandleContentBank
Address of an IHANDLE that will receive the handle of the next Content Bank.

iHandleContentBank
Handle of the last Content Bank enumerated or NULL to begin enumeration.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	No more Content Banks found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

Once a Content Bank handle is discovered, an application can pass this handle to GetNextContent to enumerate all the Content contained in the Bank.

See Also

GetContentBankName, GetNextContent, GetContentName

ISACTInterface::GetContentBankName

The **ISACTInterface::GetContentBankName** method is used to retrieve the name of the Content Bank.

```
ISACTRETURN GetContentBankName(
        char *szContentBankName,
        IHANDLE iHandleContentBank
);
```

```
ISACTRETURN GetContentBankName(
        unsigned short *szContentBankName,
        IHANDLE iHandleContentBank
);
```

Parameters

szContentBankName
Address of string to receive name of Content Bank.

iHandleContentBank
Handle of Content Bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Bad Content Bank handle
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The szContentBankName is assumed to be ISACT_MAX_NAME_LENGTH characters in length.

See Also

GetNextContent, GetContentName

ISACTInterface::GetNextContent

The **ISACTInterface::GetNextContent** method is used to enumerate all the Content in a Content Bank that has been loaded by LoadFile or LoadContentBank.

```
ISACTRETURN GetNextContent(
        IHANDLE *piHandleContent,
        IHANDLE iHandleContent,
        IHANDLE iHandleContentBank
);
```

Parameters

piHandleContent
Address of an IHANDLE that will receive the handle of the next Content in the Content Bank.

iHandleContent
Handle of the last Content enumerated or NULL to begin enumeration.

iHandleContentBank
Handle of the Content Bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	No more Content found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

After an application has retrieved the handle of a Content Bank, it can use this method to enumerate all the Content in the bank. This function only enumerates playable content.

See Also

GetContentName

ISACTInterface::GetContentName

The **ISACTInterface::GetContentName** method is used to retrieve the name of the Content.

```
ISACTRETURN GetContentName(
        char *szContentName,
        IHANDLE iHandleContent
);
```

```
ISACTRETURN GetContentName(
        unsigned short *szContentName,
        IHANDLE iHandleContent
);
```

Parameters

szContentName
Address of string to receive name of Content.

iHandleContent
Handle of Content.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Bad Content handle
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The szContentName is assumed to be ISACT_MAX_NAME_LENGTH characters in length.

See Also

GetNextContent

ISACTInterface::GetSampleBankIndex

The **ISACTInterface::GetSampleBankIndex** method is used to retrieve the Bank index that a particular Sample Bank was loaded into.

```
ISACTRETURN GetSampleBankIndex(
        unsigned long *pulBankIndex,
        const char *szSampleBankName
);
```

```
ISACTRETURN GetSampleBankIndex(
        unsigned long *pulBankIndex,
        const unsigned short *szSampleBankName
);
```

Parameters

pulBankIndex
Address of an unsigned long that will receive the Bank Index of the Sample Bank.

szSampleBankName
Name of the requested Sample Bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	Sample Bank not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The Sample Bank may have been previously loaded by LoadFile or LoadSampleBank.

See Also

LoadFile, LoadSampleBank, IsValidSampleBankIndex

ISACTInterface::IsValidSampleBankIndex

The **ISACTInterface::IsValidSampleBankIndex** method is used to determine if the given Sample Bank index contains a Sample Bank.

```
ISACTRETURN IsValidSampleBankIndex(
        unsigned long ulBankIndex
);
```

Parameters

ulBankIndex
Index of Sample Bank to validate or the predefined value ISACT_SAC_SAMPLE_BANK to access the SAC file sample bank.

Return Values

Identifier	Description
ISACT_OK	Sample Bank index is valid
ISACT_BAD_VALUE	Sample Bank index is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

None.

See Also

GetSampleBankIndex

ISACTInterface::GetSampleHandle

The **ISACTInterface::GetSampleHandle** method is used to retrieve the handle of a Sample from a Sample Bank. There are 3 methods for doing this. The first requires knowing the index of the sample. The last two require knowing the name of the sample and cover either ASCII or UNICODE formats for the name.

```
ISACTRETURN GetSampleHandle(
        IHANDLE *piSampleHandle,
        unsigned long ulSampleIndex,
        unsigned long ulBankIndex
);
```

```
ISACTRETURN GetSampleHandle(
        IHANDLE *piSampleHandle,
        const char *szSampleName,
        unsigned long ulBankIndex
);
```

```
ISACTRETURN GetSampleHandle(
        IHANDLE *piSampleHandle,
        const unsigned short *szSampleName,
        unsigned long ulBankIndex
);
```

Parameters

piSampleHandle
Address of an IHANDLE that will receive the handle of the Sample.

ulSampleIndex
Index of the Sample in the Sample Bank.

szSampleName
Name of the Sample in the Sample Bank.

ulBankIndex
Index of the Sample Bank or the predefined value ISACT_SAC_SAMPLE_BANK to access the SAC file sample bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	Sample not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

A Sample handle is used to attach the sample to an Audio Player.

See Also

LoadSampleBank, GetSampleBankIndex, IsValidSampleBankIndex, IsValidSampleHandle

ISACTInterface::IsValidSampleHandle

The **ISACTInterface::IsValidSampleHandle** method is used to verify if a Sample handle refers to a valid Sample.

```
ISACTRETURN IsValidSampleHandle(
        IHANDLE iHandleSample
);
```

Parameters

iHandleSample
Handle to validate.

Return Values

Identifier	Description
ISACT_OK	Sample handle is valid
ISACT_BAD_HANDLE	Sample handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

None.

See Also

GetSampleHandle

ISACTInterface::CacheContentSamples

The **ISACTInterface::CacheContentSamples** method is used to load all of the samples used by the content into memory.

```
ISACTRETURN CacheContentSamples(
        IHANDLE iHandleContent,
        unsigned long ulReserved = 0
);
```

Parameters

iHandleContent
Valid content handle returned from GetContentHandle.

ulReserved
Reserved for future use.

Return Values

Identifier	Description
ISACT_OK	Samples were loaded into memory
ISACT_BAD_HANDLE	The passed in content handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

This function can be used to load all of the samples required to play a piece of content into memory. The sample bank(s) required for the content must first be loaded via LoadSampleBank, LoadStreamingSampleBank, or LoadSampleBankEx.

If the SampleBank includes run-time decoding information, the run-time engine will use this information to determine whether to decode compressed audio at load-time or at run-time (play-time). If run-time decoding is enabled, then the compressed sample data will be loaded into memory and streamed (and decoded) at play-time. If run-time decoding is not enabled, then the uncompressed sample will be stored in memory, and will no longer be streamed. There is currently no way to uncache the sample without unloading the Sample Bank.

See Also

CacheStreamingSample

ISACTInterface::CacheEntitySamples

The **ISACTInterface::CacheEntitySamples** method is used to load all of the samples used by all the content referenced by the Entity into memory.

```
ISACTRETURN CacheEntitySamples(
        IHANDLE iHandleEntity,
        unsigned long ulReserved = 0
);
```

Parameters

iHandleEntity
Valid Entity handle returned from CreateEntity.

ulReserved
Reserved for future use.

Return Values

Identifier	Description
ISACT_OK	Samples were loaded into memory
ISACT_BAD_HANDLE	The passed in Entity handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

This function can be used to load all of the samples required by the Content objects that could be played by the Entity Script into memory. The sample bank(s) required for the content must first be loaded via LoadSampleBank, LoadStreamingSampleBank, or LoadSampleBankEx.

If the SampleBank includes run-time decoding information, the run-time engine will use this information to determine whether to decode compressed audio at load-time or at run-time (play-time). If run-time decoding is enabled, then the compressed sample data will be loaded into memory and streamed (and decoded) at play-time. If run-time decoding is not enabled, then the uncompressed sample will be stored in memory, and will no longer be streamed. There is currently no way to uncache the sample without unloading the Sample Bank.

ISACTInterface::CacheStreamingSample

The **ISACTInterface::CacheStreamingSample** method is used to load a sample in a streaming sample bank into memory.

```
ISACTRETURN CacheStreamingSampleBank(
        char *szSampleName,
        unsigned long ulBankIndex
);
ISACTRETURN CacheStreamingSampleBank(
        unsigned short *szSampleName,
        unsigned long ulBankIndex
);
ISACTRETURN CacheStreamingSampleBank(
        unsigned long ulSampleIndex,
        unsigned long ulBankIndex
);
ISACTRETURN CacheStreamingSampleBank(
        IHANDLE iHandleSample
);
```

Parameters

szSampleName
Name of the Sample in the Sample Bank.

ulBankIndex
Index of the Sample Bank.

iHandleSample
Handle to validate.

Return Values

Identifier	Description
ISACT_OK	Successful
ISACT_NOT_FOUND	Sample not found
ISACT_BAD_HANDLE	Sample handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

If the SampleBank includes run-time decoding information, the run-time engine will use this information to determine whether to decode compressed audio at load-time or at run-time (play-time). If run-time decoding is enabled, then the compressed sample data will be loaded into memory and streamed (and decoded) at play-time. If run-time decoding is not enabled, then the uncompressed sample will be stored in memory, and will no longer be streamed. There is currently no way to uncache the sample without unloading the Sample Bank.

See Also

GetSampleHandle, LoadSteamingSampleBank

ISACTInterface::UpdateStreamingSampleBanks

The **ISACTInterface::UpdateStreamingSampleBanks** method is used to refresh all currently playing sample streams.

```
ISACTRETURN UpdateStreamingSampleBanks(
        VOID
);
```

Parameters

None.

Return Values

Identifier	Description
ISACT_OK	Updates were successful
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

When using Streaming Sample Banks, or Sample Banks that have run-time decompression enabled, this function needs to be called at regular intervals (e.g. every frame cycle) to continually feed the playing streams. When loading Sample Banks via LoadStreamingSampleBanks do not call this function until all the banks have been loaded.

See Also

LoadStreamingSampleBank

ISACTInterface::GetSampleBankName

The **ISACTInterface::GetSampleBankName** method is used to retrieve the name of the Sample Bank loaded at the given Bank index.

```
ISACTRETURN GetSampleBankName(
        char *szSampleBankName,
        unsigned long ulBankIndex
);
```

```
ISACTRETURN GetSampleBankName(
        unsigned short *szSampleBankName,
        unsigned long ulBankIndex
);
```

Parameters

szSampleBankName
Address of string to receive name of Sample Bank.

ulBankIndex
Sample Bank index to query or the predefined value ISACT_SAC_SAMPLE_BANK to access the SAC file sample bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	Sample Bank index does not contain a Sample Bank
ISACT_BAD_VALUE	Sample Bank index out of range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The szSampleBankName is assumed to be ISACT_MAX_NAME_LENGTH characters in length.

See Also

GetSampleName

ISACTInterface::GetSampleName

The **ISACTInterface::GetSampleName** method is used to retrieve the name of the Sample at the given Bank and Sample index.

```
ISACTRETURN GetSampleName(
        char *szSampleName,
        unsigned long ulSampleIndex,
        unsigned long ulBankIndex
);
```

```
ISACTRETURN GetSampleName(
        unsigned short *szSampleName,
        unsigned long ulSampleIndex,
        unsigned long ulBankIndex
);
```

Parameters

szSampleName
Address of string to receive name of Sample.

ulSampleIndex
Index of Sample.

ulBankIndex
Index of Sample Bank or the predefined value ISACT_SAC_SAMPLE_BANK to access the SAC file sample bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_FOUND	Bank/Sample Index does not contain a Sample
ISACT_BAD_VALUE	Index out of range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The szSampleName is assumed to be ISACT_MAX_NAME_LENGTH characters in length.

See Also

GetSampleBankName

Global Functions

Audio	GetAvailableVoiceCount
	StopAllAudio
	PauseAllAudio
	RestartAllAudio
	SetHeadRoomGain
	GetHeadRoomGain
Distance	SetDistanceFactor
	GetDistanceFactor
Listener	SetListenerGain
	SetListenerPosition
	SetListenerOrientation (Vector based)
	SetListenerOrientation (Euler based)
	GetListenerPosition
	GetListenerOrientation (Vector based)
	GetListenerOrientation (Euler based)
EAX	SetEAXProperty
	GetEAXProperty

ISACTInterface::GetAvailableVoiceCount

The **ISACTInterface::GetAvailableVoiceCount** method is used to determine the number of free voices assigned to the ISACT Engine in the Audio Rendering Interface.

```
ISACTRETURN GetAvailableVoiceCount(
        long *plCount
);
```

Parameters

plCount
Address of a long variable that will receive the number of free voices.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	One or more parameters are invalid or out-of-range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

An application can use this method to determine if enough voices are available to play a particular piece of Content. To determine the number of voices required to play a specific piece of Content, the application should use GetContentVoiceCount.

See Also

GetContentVoiceCount

ISACTInterface::StopAllAudio

The **ISACTInterface::StopAllAudio** method stops all Audio Players currently playing that have Primary or Transition Content attached, that belongs to the given Group. Specifiying NULL for the Group handle will stop all Audio Players.

```
ISACTRETURN StopAllAudio(
        IHANDLE iHandleGroup
);
```

Parameters

iHandleGroup
A valid Group handle, or, NULL to affect all Audio Players.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Handle is neither NULL nor a valid Group Handle
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

This helper function can be used to stop all audio output by passing in a Group handle of NULL.

See Also

PauseAllAudio, RestartAllAudio

ISACTInterface::PauseAllAudio

The **ISACTInterface::PauseAllAudio** method will pause all the Audio Players currently playing that have Primary or Transition Content attached, that belongs to the given Group. Specifiying NULL for the Group handle will pause all the Audio Players.

```
ISACTRETURN PauseAllAudio(
        IHANDLE iHandleGroup
);
```

Parameters

iHandleGroup
A valid Group handle, or, NULL to affect all Audio Players.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Handle is neither NULL nor a valid Group Handle
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

This helper function can be used to pause all audio output by passing in a Group handle of NULL. Audio output can be resumed using RestartAllAudio with a NULL Group handle.

See Also

StopAllAudio, RestartAllAudio

ISACTInterface::RestartAllAudio

The **ISACTInterface::RestartAllAudio** will resume playback of all the paused Audio Players that have Primary or Transition Content attached, that belongs to the given Group. Specifiying NULL for the Group handle will resume playback of all the paused Audio Players.

```
ISACTRETURN PauseAllAudio(
        IHANDLE iHandleGroup
);
```

Parameters

iHandleGroup
A valid Group handle, or, NULL to affect all Audio Players.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Handle is neither NULL nor a valid Group Handle
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

This helper function can be used to resume playback of all audio output by passing in a Group handle of NULL.

See Also

StopAllAudio, PauseAllAudio

ISACTInterface::SetHeadRoomGain

The **ISACTInterface::SetHeadRoomGain** method is used to attenuate all Audio Content. Unlike a global volume, individual ISACT Content objects can have an optional Head Room Boost that can be used to remove the attenation effect of the global head-room gain. This allows the sound designer to enable certain sounds (e.g explosions and other loud noises) to stand out in the final mix.

```
ISACTRETURN SetHeadRoomGain(
        float flGain
);
```

Parameters

flGain

The amount of attenuation to apply to all Audio Content (0.0 to 1.0).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	Parameter is out of range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

To give all Audio Content a 6dB headroom the Head Room Gain should be set to 0.5. All sounds (without a Head Room Boost) will play at half-volume. To give the Audio Content a 12dB headroom, the Head Room Gain should be set to 0.25.

See Also

GetHeadRoomGain

ISACTInterface::GetHeadRoomGain

The **ISACTInterface::GetHeadRoomGain** method is used to retrieve the current value of the Global Head Room Gain.

```
ISACTRETURN GetHeadRoomGain(
        float *pflGain
);
```

Parameters

pflGain
Address of variable to receive the current Head Room Gain.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	Bad pointer
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The default value of Head Room Gain is 1.0 which gives no attenuation, and therefore no headroom.

See Also

SetHeadRoomGain

ISACTInterface::SetDistanceFactor

The **ISACTInterface::SetDistanceFactor** method is used to convert the applications units into metres. If the applications units are in feet, the Distance Factor should be set to 0.3048. All Listener Positions, and Player and Entity Positions and Velocities will be internally multiplied by the Distance Factor before being combined with the Content's co-ordinates.

```
ISACTRETURN SetDistanceFactor(
        float flDistanceFactor
);
```

Parameters

flDistanceFactor
New value for the Distance Factor (>= FLT_MIN).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	Bad value for Distance Factor
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

It is recommended to set the Distance Factor before any Content is played. Adjusting the Distance Factor when Content is playing, may result in temporary audio artifacts while 3D positions and velocities are being updated.

See Also

GetDistanceFactor

ISACTInterface::GetDistanceFactor

The **ISACTInterface::GetDistanceFactor** method is used to retrieve the current value of the Distance Factor.

```
ISACTRETURN GetDistanceFactor(
        float *pflDistanceFactor
);
```

Parameters

pflDistanceFactor
Address of a floating point variable to receive the Distance Factor.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Default value for the Distance Factor is 1.0.

See Also

SetDistanceFactor

ISACTInterface::SetListenerGain

The **ISACTInterface::SetListenerGain** method controls a global master volume level of all ISACT output.

```
ISACTRETURN SetListenerGain(
        float fGain
);
```

Parameters

fGain

New global volume level between 0 (silence) to 1 (full volume).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

An application can use this method to quickly mute audio output if the application loses window focus, and restore it, when the application re-gains the focus.

ISACTInterface::SetListenerPosition

The **ISACTInterface::SetListenerPosition** method sets the position of the Listener in 3D space.

```
ISACTRETURN SetListenerPosition(
        ISACTVector &vPosition
);
```

Parameters

vPosition
New position for the Listener.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

See Also

GetListenerPosition

ISACTInterface::SetListenerOrientation (Vector based)

The **ISACTInterface::SetListenerOrientation** method sets the orientation of the Listener. Two orientation vectors (front and up directions) are necessary to control the listener's orientation.

```
ISACTRETURN SetListenerOrientation(
        ISACTVector &vFront,
        ISACTVector &vUp
);
```

Parameters

vFront
New forwards, or 'at' direction.

vUp
New up direction.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The 'Front' and 'Up' vectors should be orthogonal.

Listener orientation can also be set using Euler angles using SetListenerOrientation (Euler based).

See Also

GetListenerOrientation (Vector based)

ISACTInterface::SetListenerOrientation (Euler based)

The **ISACTInterface::SetListenerOrientation** method sets the orientation of the Listener.

```
ISACTRETURN SetListenerOrientation(
        float fAzimuth,
        float fHeight,
        float fRoll
);
```

Parameters

fAzimuth

fHeight

fRoll

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

The Listener's orientation can also be set using vectors using SetListenerOrientation (Vector based).

See Also

GetListenerOrientation (Euler based)

ISACTInterface::GetListenerPosition

The **ISACTInterface::GetListenerPosition** method retrieves the position of the Listener in 3D space.

```
ISACTRETURN GetListenerPosition(
        ISACTVector *pvPosition
);
```

Parameters

pvPosition
Address of an ISACTVector that will receive the position of the Listener.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

SetListenerPosition

ISACTInterface::GetListenerOrientation (Vector based)

The **ISACTInterface::GetListenerOrientation** method retrieves the orientation of the Listener.

```
ISACTRETURN GetListenerOrientation(
        ISACTVector *pvFront,
        ISACTVector *pvUp
);
```

Parameters

pvFront
Address of an ISACTVector that receives the forwards or ‘at’ direction.

pvUp
Address of an ISACTVector that receives the up direction.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Listener orientation can also be retrieved in Euler angles using GetListenerOrientation (Euler based).

See Also

SetListenerOrientation (Vector based)

ISACTInterface::GetListenerOrientation (Euler based)

The **ISACTInterface::GetListenerOrientation** method retrieves the orientation of the Listener in Euler angles.

```
ISACTRETURN GetListenerOrientation(
        float *pflAzimuth,
        float *pflHeight,
        float *pflRoll
);
```

Parameters

pflAzimuth
Address of variable that receives the azimuth angle

pflHeight
Address of variable that receives the height angle

pflRoll
Address of variable that receives that roll angle

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The Listener's orientation can also be retrieved as two vectors ('at' and 'up') using GetListenerOrientation (Vector based).

See Also

SetListenerOrientation (Euler based)

ISACTInterface::SetEAXProperty

The **ISACTInterface::SetEAXProperty** method can be used to set global EAX parameters or EAX Source parameters on the given Player or Entity.

```
ISACTRETURN SetEAXProperty(
        IHANDLE iHandle,
        GUID &Guid,
        unsigned long ulProperty,
        void *pData,
        unsigned long ulDataSize
);
```

Parameters

iHandle
Valid Player handle returned from CreateAudioPlayer, or a valid Entity handle returned from CreateEntity, or NULL for setting global EAX properties.

Guid
GUID that identifies the EAX object.

ulProperty
Enumeration value of the EAX property to set.

pData
Pointer to the data to set.

ulDataSize
Size of data pointed to by pData.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player or Entity handle
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

None.

See Also

GetEAXProperty

ISACTInterface::GetEAXProperty

The **ISACTInterface::GetEAXProperty** method can be used to retrieve global EAX properties or EAX Source properties from the given Player or Entity.

```
ISACTRETURN GetEAXProperty(
        IHANDLE iHandle,
        GUID &Guid,
        unsigned long ulProperty,
        void *pData,
        unsigned long ulDataSize
);
```

Parameters

iHandle
Valid Player handle returned from CreateAudioPlayer, or a valid Entity handle returned from CreateEntity, or NULL for retrieving global EAX properties.

Guid
GUID that identifies the EAX object.

ulProperty
Enumeration value of the EAX property to retrieve.

pData
Pointer to the location where the data should be written.

ulDataSize
Size of data pointed to by pData.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player or Entity handle
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

None.

See Also

SetEAXProperty

Audio Player Functions

Handle manipulation	CreateAudioPlayer
	DestroyAudioPlayer
	IsValidPlayerHandle
Content	AttachPlayerPrimaryContent
	AttachPlayerTransitionContent
	AttachPlayerRPC
	ReleasePlayerPrimaryContent
	ReleasePlayerTransitionContent
	ReleasePlayerRPC
Synchronization	SetMasterAudioPlayer
	GetMasterAudioPlayer
Player Status	PlayAudioPlayer
	TransitionAudioPlayer
	PauseAudioPlayer
	RestartAudioPlayer
	StopAudioPlayer
	GetAudioPlayerStatus
	GetAudioPlayerMetricTime
	GetAudioPlayerRealTime
Properties	SetAudioPlayerMode
	SetAudioPlayerGain
	SetAudioPlayerPitch
	SetAudioPlayerPosition
	SetAudioPlayerOrientation (Vector based)
	SetAudioPlayerOrientation (Euler based)
	SetAudioPlayerVelocity
	SetAudioPlayerRPCLevel
	GetAudioPlayerMode
	GetAudioPlayerGain
	GetAudioPlayerPitch
	GetAudioPlayerPosition
	GetAudioPlayerOrientation (Vector based)
	GetAudioPlayerOrientation (Euler based)
	GetAudioPlayerVelocity
	GetAudioPlayerRPCLevel
	GetAudioPlayerMaxGain

ISACTInterface::CreateAudioPlayer

The **ISACTInterface::CreateAudioPlayer** method creates a new audio player object and returns a handle which can be used in other methods that require an audio player handle.

```
ISACTRETURN CreateAudioPlayer (
        IHANDLE *piHandle
);
```

Parameters

piHandle
Address of an IHANDLE data type that will receive the new player handle if successful.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_MEMORY_ERROR	An error occurred attempting to allocate needed memory.

Remarks

ISACT_BAD_VALUE will only be returned if the passed in piHandle parameter is NULL. If piHandle is not NULL it is assumed to be a valid pointer and no other validation is done.

See Also

DestroyAudioPlayer, IsValidPlayerHandle

ISACTInterface::DestroyAudioPlayer

The **ISACTInterface::DestroyAudioPlayer** method destroys a previously created audio player by releasing all memory associated with the passed in player handle.

```
ISACTRETURN DestroyAudioPlayer (
        IHANDLE iHandle
);
```

Parameters

iHandle
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

This method uses the IsValidPlayerHandle method to validate the passed in player handle. If the handle is not valid no action is taken and the return identifier is ISACT_BAD_HANDLE. Upon successful completion the passed in handle is no longer valid and should not be used again.

Any content associated with the player is simply disassociated before the player is destroyed.

See Also

CreateAudioPlayer, IsValidPlayerHandle

ISACTInterface::IsValidPlayerHandle

The **ISACTInterface::IsValidPlayerHandle** method can be used to verify if a handle is a player handle and is still valid.

```
ISACTRETURN IsValidPlayerHandle (
        IHANDLE iHandle
);
```

Parameters

iHandle
IHANDLE data type to be verified.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

If the passed in handle is a valid player handle the return identifier is ISACT_OK. If it is not a valid handle the return identifier is ISACT_BAD_HANDLE.

See Also

CreateAudioPlayer, DestroyAudioPlayer

ISACTInterface::AttachPlayerRPC

The **ISACTInterface::AttachPlayerRPC** method is used to attach a Real-Time Parameter Controller to the player, and set the RPC parameter level.

```
ISACTRETURN AttachPlayerRPC(
        IHANDLE iHandlePlayer,
        const char *szRPCName,
        IHANDLE iHandleContentBank,
        float flRPCLevel
);
```

```
ISACTRETURN AttachPlayerRPC(
        IHANDLE iHandlePlayer,
        const unsigned short *szRPCName,
        IHANDLE iHandleContentBank,
        float flRPCLevel
);
```

Parameters

IHandlePlayer
Valid player handle returned from CreateAudioPlayer.

szRPCName
Name of the requested RPC.

IHandleContentBank
Valid content bank handle or NULL to use the SAC file content bank.

flRPCLevel
Initial level of RPC parameter, between 0 and 1

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	A specified handle is not valid

Remarks

RPC's attached to a Player will affect all the voices used by the primary (and transitioning) content. Real-time Parameter Controllers can affect Pitch, Volume, Direct, Direct HF, and the Send and Send HF levels to each EAX FX Slot. The parameters affected, and how they respond to the input RPC level are determined by the RPC author. After an RPC has been attached, the RPC level can be controlled using SetAudioPlayerRPCLevel.

See Also

CreateAudioPlayer, ReleasePlayerRPC

ISACTInterface::AttachPlayerPrimaryContent

The **ISACTInterface::AttachPlayerPrimaryContent** method is used to set the piece of audio content the player will be play when the PlayAudioPlayer method is used.

```
ISACTRETURN AttachPlayerPrimaryContent (
        IHANDLE iHandlePlayer,
        IHANDLE iHandleContent
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

iHandleContent
Valid content handle returned from GetContentHandle.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	A specified handle is not valid.
ISACT_MEMORY_ERROR	An error occurred attempting to allocate needed memory.

Remarks

Before the content is attached to the player, all currently playing content is immediately stopped and any voices allocated for primary and/or transition content are released. Then any currently attached primary and/or transition content are released from the player and any memory allocated for the content is released.

When attaching the content to the player a certain amount of memory will be allocated for use by the content. The size of this memory is dependent on the type of content being attached. Attaching a piece of content to a player does not allocate any voices. Once attached the content is reset to the beginning.

See Also

CreateAudioPlayer, GetContentHandle, AttachPlayerTransitionContent, ReleasePlayerPrimaryContent

ISACTInterface::AttachPlayerTransitionContent

The **ISACTInterface::AttachPlayerTransitionContent** method is used to set the piece of content the player will transition to play when the TransitionAudioPlayer method is used.

```
ISACTRETURN AttachPlayerTransitionContent (
        IHANDLE iHandlePlayer,
        IHANDLE iHandleContent
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

iHandleContent
Valid content handle returned from GetContentHandle.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	A specified handle is not valid
ISACT_MEMORY_ERROR	An error occurred attempting to allocate needed memory
ISACT_IN_TRANSITION	The method could not continue because the player was in a transition

Remarks

Add remarks here.

See Also

AttachPlayerPrimaryContent, ReleasePlayerTransitionContent

ISACTInterface::ReleasePlayerPrimaryContent

The **ISACTInterface::ReleasePlayerPrimaryContent** method is used to remove the Primary Content attached to a Player.

```
ISACTRETURN ReleasePlayerPrimaryContent (
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

This method will stop the Player if it currently playing content.

See Also

CreateAudioPlayer, AttachPlayerPrimaryContent

ISACTInterface::ReleasePlayerTransitionContent

The **ISACTInterface::ReleasePlayerTransitionContent** method

```
ISACTRETURN ReleasePlayerTransitionContent (
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Add remarks here.

See Also

CreateAudioPlayer, AttachPlayerTransitionContent

ISACTInterface::ReleasePlayerRPC

The **ISACTInterface::ReleasePlayerRPC** method is used to remove a previously attached Real-Time Parameter Controller.

```
ISACTRETURN ReleasePlayerTransitionContent (
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Removing an RPC, will remove any effect the RPC had on the Player's Pitch, Volume, Direct, Direct HF, and Send and Send HF to each EAX Effect Slot.

See Also

CreateAudioPlayer, AttachPlayerRPC

ISACTInterface::SetMasterAudioPlayer

The **ISACTInterface::SetMasterAudioPlayer** method.

```
ISACTRETURN SetMasterAudioPlayer(
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer or NULL.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error

Remarks

Remarks.

See Also

CreateAudioPlayer

ISACTInterface::GetMasterAudioPlayer

The **ISACTInterface::GetMasterAudioPlayer** method.

```
ISACTRETURN GetMasterAudioPlayer(
        IHANDLE *piHandlePlayer,
);
```

Parameters

piHandlePlayer
Address of an IHANDLE that will receive the handle of the Master Player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Remarks.

See Also

CreateAudioPlayer

ISACTInterface::PlayAudioPlayer

The **ISACTInterface::PlayAudioPlayer** method is used to start the playback of the primary content attached to the player.

```
ISACTRETURN PlayAudioPlayer (
        IHANDLE iHandlePlayer,
        long lLoopCount  = 0,
        IHANDLE iHandleMarker = 0
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

lLoopCount
Number of times to repeat or loop the content, or the flag value LOOP_INFINITE to continually loop.

iHandleMarker
Valid marker handle returned from GetContentMarker or 0 if playback is to start from the beginning of the content.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	A specified handle is not valid

Remarks

Add remarks here.

See Also

CreateAudioPlayer

ISACTInterface::TransitionAudioPlayer

The **ISACTInterface::TransitionAudioPlayer** method is used to start the playback of the transition content attached to the player.

```
ISACTRETURN TransitionAudioPlayer (
        IHANDLE iHandlePlayer,
        const char *szTransitionName,
        IHANDLE iHandleContentBank,
        long lLoopCount  = 0,
        IHANDLE iHandleMarker = 0
);
```

```
ISACTRETURN TransitionAudioPlayer (
        IHANDLE iHandlePlayer,
        const unsigned short *szTransitionName,
        IHANDLE iHandleContentBank,
        long lLoopCount  = 0,
        IHANDLE iHandleMarker = 0
);
```

```
ISACTRETURN TransitionAudioPlayer (
        IHANDLE iHandlePlayer,
        STransition *pTransition,
        long lLoopCount  = 0,
        IHANDLE iHandleMarker = 0
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

szTransitionName
Name of the requested transition.

iHandleContentBank
Valid content bank handle or NULL to use the SAC file content bank.

pTransition
Pointer to a valid STransition structure.

lLoopCount
Number of times to repeat or loop the content, or the flag value LOOP_INFINITE to continually loop.

iHandleMarker
Valid marker handle returned from GetContentMarker or 0 if playback is to start from the beginning of the content.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error

ISACT_BAD_HANDLE	A specified handle is not valid

Remarks

Add remarks here.

See Also

Transitioning, AttachPlayerTransitionContent

ISACTInterface::PauseAudioPlayer

The **ISACTInterface::PauseAudioPlayer** method pauses the content playing on this Player.

```
ISACTRETURN PauseAudioPlayer (
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Add remarks here.

See Also

RestartAudioPlayer

ISACTInterface::RestartAudioPlayer

The **ISACTInterface::RestartAudioPlayer** method

```
ISACTRETURN RestartAudioPlayer (
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Add remarks here.

See Also

PauseAudioPlayer

ISACTInterface::StopAudioPlayer

The **ISACTInterface::StopAudioPlayer** method

```
ISACTRETURN StopAudioPlayer (
        IHANDLE iHandlePlayer,
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Stops the Content attached to the player. Note: Stopping a Player with a SoundQueue attached may not stop immediately. SoundQueue Content can contain a looped section that continues to loop until StopAudioPlayer is called. When this happens the looped section no longer loops and the SoundQueue will play through to end of the queued content. During this time, a call to get the status of the Player will return PLAYER_RELEASE, instead of PLAYER_STOPPED. A second call to StopAudioPlayer will force the SoundQueue to immediately halt playback.

See Also

CreateAudioPlayer, GetAudioPlayerStatus

ISACTInterface::GetAudioPlayerStatus

The **ISACTInterface::GetAudioPlayerStatus** method

```
ISACTRETURN GetAudioPlayerStatus (
        IHANDLE iHandlePlayer,
        SPlayerStatus *pPlayerStatus
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pPlayerStatus
A pointer to a valid SPlayerStatus structure to be filled with the Player's current status information.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Add remarks here.

See Also

CreateAudioPlayer, SPlayerStatus

ISACTInterface::GetAudioPlayerMetricTime

The **ISACTInterface::GetAudioPlayerMetricTime** method

```
ISACTRETURN GetAudioPlayerMetricTime(
        IHANDLE iHandlePlayer,
        SPlayerMetricTime *pPlayerMetricTime
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pPlayerMetricTime
A pointer to a valid SPlayerMetricTime structure, which will be filled with the Player's current position in Metric Time.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Add remarks here.

See Also

CreateAudioPlayer, GetAudioPlayerRealTime, SPlayerMetricTime

ISACTInterface::GetAudioPlayerRealTime

The **ISACTInterface::GetAudioPlayerRealTime** method

```
ISACTRETURN GetAudioPlayerRealTime(
        IHANDLE iHandlePlayer,
        SPlayerRealTime *pPlayerRealTime
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pPlayerRealTime
A pointer to a valid SPlayerRealTime structure to be filled with the Player's current position in Real Time.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Add remarks here.

See Also

CreateAudioPlayer, GetAudioPlayerMetricTime, SPlayerRealTime

ISACTInterface::SetAudioPlayerMode

The **ISACTInterface::SetAudioPlayerMode** method sets the 3D Audio Head-Relative mode. When Head-Relative mode is enabled, the sound's position, velocity and orientation are relative to the listener. If the listener changes position, velocity or orientation the Player is automatically updated to maintain the same relative values.

```
ISACTRETURN SetAudioPlayerMode(
        IHANDLE iHandlePlayer,
        unsigned long ulMode
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

ulMode
TRUE for Head-Relative, FALSE for normal processing.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

None.

See Also

CreateAudioPlayer, GetAudioPlayerMode, SetAudioPlayerPosition, SetAudioPlayerVelocity, SetAudioPlayerOrientation

ISACTInterface::SetAudioPlayerGain

The **ISACTInterface::SetAudioPlayerGain** method sets the gain or volume of the player. Content playing on this player will be offset by this volume.

```
ISACTRETURN SetAudioPlayerGain(
        IHANDLE iHandlePlayer,
        float flGain,
        unsigned long ulFadeTime = 0
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

flGain
New gain level for player (0 means no volume, 1 means full volume).

ulFadeTime
Time in milliseconds to fade the volume to the new level (default is 0).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

All gains are clamped between 0 and 1.

See Also

CreateAudioPlayer, GetAudioPlayerGain

ISACTInterface::SetAudioPlayerPitch

The **ISACTInterface::SetAudioPlayerPitch** method sets the pitch of the player.

```
ISACTRETURN SetAudioPlayerPitch(
        IHANDLE iHandlePlayer,
        float flPitch
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

flGain
New pitch level for player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

Currently pitch is calmped between 0.001 to 2.0.

See Also

CreateAudioPlayer, GetAudioPlayerPitch

ISACTInterface::SetAudioPlayerPosition

The **ISACTInterface::SetAudioPlayerPosition** method sets the position of the Player in 3D space. Content playing on this player will be offset by this position.

```
ISACTRETURN SetAudioPlayerPosition(
        IHANDLE iHandlePlayer,
        ISACTVector &vPosition
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

vPosition
New position for Player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

None.

See Also

CreateAudioPlayer, GetAudioPlayerPosition

ISACTInterface::SetAudioPlayerOrientation (Vector based)

The **ISACTInterface::SetAudioPlayerOrientation** method sets the orientation of the Player. Content playing on this player will be rotated by this orientation. As the Content attached to a Player can contain multiple voices with different positions, it is necessary to provide two orientation vectors (front and up directions) for correct rotation of the Content.

```
ISACTRETURN SetAudioPlayerOrientation(
        IHANDLE iHandlePlayer,
        ISACTVector &vFront,
        ISACTVector &vUp
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

vFront
New forwards, or 'at' direction.

vUp
New up direction.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

The 'Front' and 'Up' vectors should be orthogonal.

Player orientation can also be set using Euler angles using SetAudioPlayerOrientation (Euler based).

See Also

CreateAudioPlayer, GetAudioPlayerOrientation (Vector based)

ISACTInterface::SetAudioPlayerOrientation (Euler based)

The **ISACTInterface::SetAudioPlayerOrientation** method sets the orientation of the Player. Content playing on this player will be rotated by this orientation.

```
ISACTRETURN SetAudioPlayerOrientation(
        IHANDLE iHandlePlayer,
        float flAzimuth,
        float flHeight,
        float flRoll
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

flAzimuth

flHeight

flRoll

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

The Player's orientation can also be set using vectors using SetAudioPlayerOrientation (Vector based).

See Also

CreateAudioPlayer, GetAudioPlayerOrientation (Euler based)

ISACTInterface::SetAudioPlayerVelocity

The **ISACTInterface::SetAudioPlayerVelocity** method sets the velocity of the Player in 3D space. Content playing on this player will be offset by this velocity.

```
ISACTRETURN SetAudioPlayerVelocity(
        IHANDLE iHandlePlayer,
        ISACTVector &vVelocity
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

vVelocity
New velocity for Player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

None.

See Also

CreateAudioPlayer, GetAudioPlayerVelocity

ISACTInterface::SetAudioPlayerRPCLevel

The **ISACTInterface::SetAudioPlayerRPCLevel** method updates the Real-Time Parameter Controller value for the attached RPC. If an RPC is attached, then the Player's Pitch, Volume, Direct, Direct HF, and Send and Send HF levels to all of the EAX FX Slots, may be affected depending upon the RPC control data.

```
ISACTRETURN SetAudioPlayerRPCLevel(
        IHANDLE iHandlePlayer,
        float flLevel
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

flLevel
New RPC level for Player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

The RPC level is clamped between 0 and 1.

See Also

CreateAudioPlayer, GetAudioPlayerRPCLevel

ISACTInterface::GetAudioPlayerMode

The **ISACTInterface::GetAudioPlayerMode** method retrieves the 3D Audio Head-Relative mode of the specified Player.

```
ISACTRETURN GetAudioPlayerMode(
        IHANDLE iHandlePlayer,
        unsigned long *pulMode
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pulMode
Address of a variable that will be set to TRUE if Head-Relative mode is enabled, or FALSE if not.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateAudioPlayer, SetAudioPlayerMode

ISACTInterface::GetAudioPlayerGain

The **ISACTInterface::GetAudioPlayerGain** method retrieves the gain or volume level of the Player.

```
ISACTRETURN GetAudioPlayerGain(
        IHANDLE iHandlePlayer,
        float *pflGain
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pvPosition
Address of a float that will receive the gain level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

None.

See Also

CreateAudioPlayer, SetAudioPlayerGain

ISACTInterface::GetAudioPlayerPitch

The **ISACTInterface::GetAudioPlayerPitch** method retrieves the pitch level of the Player.

```
ISACTRETURN GetAudioPlayerPitch(
        IHANDLE iHandlePlayer,
        float *pflPitch
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pvPosition
Address of a float that will receive the pitch level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle

Remarks

None.

See Also

CreateAudioPlayer, SetAudioPlayerPitch

ISACTInterface::GetAudioPlayerPosition

The **ISACTInterface::GetAudioPlayerPosition** method retrieves the position of the Player in 3D space.

```
ISACTRETURN GetAudioPlayerPosition(
        IHANDLE iHandlePlayer,
        ISACTVector *pvPosition
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pvPosition
Address of an ISACTVector that will receive the position of the Player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateAudioPlayer, SetAudioPlayerPosition

ISACTInterface::GetAudioPlayerOrientation (Vector based)

The **ISACTInterface::GetAudioPlayerOrientation** method retrieves the orientation of the Player. As the Content attached to a Player can contain multiple voices with different positions, it is necessary to provide two orientation vectors (front and up directions) for correct rotation of the Content.

```
ISACTRETURN GetAudioPlayerOrientation(
        IHANDLE iHandlePlayer,
        ISACTVector *pvFront,
        ISACTVector *pvUp
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pvFront
Address of an ISACTVector that receives the forwards or 'at' direction.

pvUp
Address of an ISACTVector that receives the up direction.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

Player orientation can also be retrieved in Euler angles using GetAudioPlayerOrientation (Euler based).

See Also

CreateAudioPlayer, SetAudioPlayerOrientation (Vector based)

ISACTInterface::GetAudioPlayerOrientation (Euler based)

The **ISACTInterface::GetAudioPlayerOrientation** method retrieves the orientation of the Player in Euler angles.

```
ISACTRETURN GetAudioPlayerOrientation(
        IHANDLE iHandlePlayer,
        float *pflAzimuth,
        float *pflHeight,
        float *pflRoll
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pflAzimuth
Address of variable that receives the azimuth angle

pflHeight
Address of variable that receives the height angle

pflRoll
Address of variable that receives that roll angle

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The Player's orientation can also be retrieved as two vectors ('at' and 'up') using GetAudioPlayerOrientation (Vector based).

See Also

CreateAudioPlayer, SetAudioPlayerOrientation (Euler based)

ISACTInterface::GetAudioPlayerVelocity

The **ISACTInterface::GetAudioPlayerVelocity** method retrieves the velocity of the Player in 3D space.

```
ISACTRETURN GetAudioPlayerVelocity(
        IHANDLE iHandlePlayer,
        ISACTVector *pvVelocity
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pvVelocity
Address of an ISACTVector variable that receives the Player's velocity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateAudioPlayer, SetAudioPlayerVelocity

ISACTInterface::GetAudioPlayerRPCLevel

The **ISACTInterface::GetAudioPlayerRPCLevel** method retrieves the current level of the Real-Time Parameter Controller attached to the Player.

```
ISACTRETURN GetAudioPlayerRPCLevel(
        IHANDLE iHandlePlayer,
        float *pflLevel
);
```

Parameters

iHandlePlayer
Valid player handle returned from CreateAudioPlayer.

pflLevel
Address of a floating point variable that will receive the RPC level of the Player.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Player handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateAudioPlayer, SetAudioPlayerRPCLevel

ISACTInterface::GetAudioPlayerMaxGain

The **ISACTInterface::GetAudioPlayerMaxGain** method is used to retrieve the loudest gain that any piece of Content attached to the Audio Player would have at a given 3D position.

```
ISACTRETURN GetAudioPlayerMaxGain(
        IHANDLE iHandle,
        float *pflGain
);
```

Parameters

iHandle
Valid Audio Player handle returned from CreateAudioPlayer.

pflGain
Address of a floating point variable that receives the Player's Max gain level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The Max Gain is calculated by finding the loudest maximum gain of each piece of Content attached to the Audio Player. The MaxGain calculation takes into account distance attenuation (if any), and content gain (content level and sub-mix level). It does not include gain adjustments that are part of the content itself (e.g volume randomization in a Sound Event, or gain levels in a Sequence), or modifiers on the Audio Player's level (e.g Audio Player Gain, or Transitions).

See Also

CreateEntity, GetContentMaxGain, GetEntityMaxGain

Audio Entity Functions

Handle manipulation	CreateEntity
	DestroyEntity
	IsValidEntityHandle
Entity manipulation	ActivateEntity
	IsEntityPlaying
	ResetEntityLocalVars
	ResetEntityGlobalVars
	SuspendEntity
	ProcessEntity
	SuspendAllEntities
	ProcessAllEntities
	SetEntityLocalVarState (String based)
	SetEntityGlobalVarState (String based)
	SetEntityLocalVarState (Index based)
	SetEntityGlobalVarState (Index based)
Properties	SetEntityMasterRPCLevel
	SetEntityRPCLevel
	SetEntityMode
	SetEntityPosition
	SetEntityOrientation (Vector based)
	SetEntityOrientation (Euler based)
	SetEntityVelocity
	SetEntityPitch
	SetEntityGain
	GetEntityLocalVarState (String based)
	GetEntityGlobalVarState (String based)
	GetEntityLocalVarState (Index based)
	GetEntityGlobalVarState (Index based)
	GetEntityMasterRPCLevel
	GetEntityRPCLevel
	GetEntityMode
	GetEntityPosition
	GetEntityOrientation (Vector based)
	GetEntityOrientation (Euler based)
	GetEntityVelocity
	GetEntityPitch
	GetEntityGain
	GetEntityMaxGain
Query	GetNumLocalVars
	GetNumGlobalVars
	GetNumLocalVarStates
	GetNumGlobalVarStates
	GetLocalVarName
	GetGlobalVarName
	GetLocalVarStateName
	GetGlobalVarStateName

ISACTInterface::CreateEntity

The **ISACTInterface::CreateEntity** method creates an instance of the given SoundEntity. If successful a handle is returned which is used to identify this entity instance in all the Entity methods. At creation time, the Entity has all its local variables set to their default states, and is in the in-active state. A separate call to ActivateEntity is used to activate the Entity to enable state machine processing.

```
ISACTRETURN CreateEntity(
        IHANDLE *piHandle,
        const char *szEntityName,
        IHANDLE iHandleContentBank
);
```

```
ISACTRETURN CreateEntity(
        IHANDLE *piHandle,
        const unsigned short *szEntityName,
        IHANDLE iHandleContentBank
);
```

Parameters

piHandle

Address of an IHANDLE data type that will receive the Entity handle if successful.

szEntityName

The name of the SoundEntity content object, which will be used to create the Entity instance.

iHandleContentBank

Handle of the Content Bank containing the SoundEntity content object, or the predefined value ISACT_SAC_CONTENT_BANK to access the SAC file content bank, or the predefined value ISACT_SEARCH_CONTENT_BANKS to search all of the currently loaded content banks for the content name.
.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_MEMORY_ERROR	An error occurred attempting to allocate needed memory.
ISACT_BAD_CONTENT	An unknown SoundEntity name was passed in.
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

ISACT_BAD_CONTENT is returned if *szSoundEntity* does not refer to a SoundEntity content object. When using ISACT_SEARCH_CONTENT_BANKS the first found occurance of the content name is returned.

See Also

DestroyEntity, ActivateEntity

ISACTInterface::DestroyEntity

The **ISACTInterface::DestroyEntity** method destroys a previously created Entity by release all memory associated with the passed in Entity handle.

```
ISACTRETURN DestroyEntity(
        IHANDLE iHandle
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Entity handle

Remarks

All the Players in the specified Entity will be stopped, and their contents released. All Local Variables states will be lost.

See Also

CreateEntity

ISACTInterface::IsValidEntityHandle

The **ISACTInterface::IsValidEntityHandle** method can be used to verify if a handle is an Entity handle and is still valid.

```
ISACTRETURN IsValidEntityHandle(
        IHANDLE iHandle
);
```

Parameters

iHandle
IHANDLE data type to be verified.

Return Values

Identifier	Description
ISACT_OK	Entity handle is valid
ISACT_BAD_HANDLE	Entity handle is not valid
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

If the passed in handle is a valid Entity handle the return identifier is ISACT_OK. If it is not a valid handle the return identifier is ISACT_BAD_HANDLE.

See Also

CreateEntity, DestroyEntity

ISACTInterface::IsEntityPlaying

The **ISACTInterface::IsEntityPlaying** method can be used to verify if any of the players associated with the entity are playing.

```
ISACTRETURN IsEntityPlaying(
        IHANDLE iHandle,
        bool *pbPlaying
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pbPlaying
Pointer to a boolean variable that will be set to reflect the playing state.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Entity handle
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

If any of the players associated with the Entity are playing, the boolean variable pointed to by pbPlaying will be set to true. If the function returns anything other than ISACT_OK the state of the boolean variable should be ignored. A player is considered playing if it is not in the PLAYER_STOPPED state.

See Also

CreateEntity, DestroyEntity, PLAYER_STATUS

ISACTInterface::ActivateEntity

The **ISACTInterface::ActivateEntity** method is used to enable / disable processing on the given Entity. Activating an Entity causes the state machine to be run using the current states of the global variables and the Entity's local variables. De-activating an Entity stops all the Entity's players.

```
ISACTRETURN ActivateEntity(
        IHANDLE iHandle,
        bool bActivate
);
```

Parameters

iHandle

Valid Entity handle returned from CreateEntity.

bActivate

Boolean variable indicating if the Entity should be activated or de-activated.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	The specified handle is not a valid Entity handle

Remarks

Setting a variable to a particular state on an active Entity will result in all the Actions that reference that variable being evaluated. Setting a variable to a state on an in-active Entity simply stores the new state of the variable, but no Actions are evaluated.

See Also

CreateEntity

ISACTInterface::ResetEntityLocalVars

The **ISACTInterface::ResetEntityLocalVars** method sets all the Entity's Local Variables to their default states. If the Entity is active all Actions that reference a local variable are evaluated.

```
ISACTRETURN ResetEntityLocalVars(
        IHANDLE iHandle
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle

Remarks

A separate function call ResetEntityGlobalVars is used to reset the state of the global variables.

See Also

CreateEntity, ResetEntityGlobalVars

ISACTInterface::ResetEntityGlobalVars

The **ISACTInterface::ResetEntityGlobalVars** method sets all the Global Variables in the given Content Bank to their default states. All the Actions in all the active Entities that reference a global variable are evaluated.

```
ISACTRETURN ResetEntityGlobalVars(
        IHANDLE iHandleContentBank
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the global variables.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank handle

Remarks

A separate function call, ResetEntityLocalVars is used to reset the state of the local variables.

See Also

ResetEntityLocalVars

ISACTInterface::SuspendEntity

The **ISACTInterface::SuspendEntity** method disables Action processing on the Entity. Subsequent calls to set variable states store the new states, but no Actions are evaluated. The SuspendEntity method can be used as an optimization when the application needs to set multiple variable states without evaluating affected Actions after each call. After the application has set all the variables to their new states, the Entity can be resumed by calling ProcessEntity which evaluates all the Actions that reference a variable whose state has been modified since the Entity was suspended.

```
ISACTRETURN SuspendEntity(
        IHANDLE iHandle
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle

Remarks

Unlike the ActivateEntity method, SuspendEntity does not stop any of the Entity's Players.

All Entities can be suspended in one call using SuspendAllEntities.

See Also

CreateEntity, ProcessEntity, ActivateEntity

ISACTInterface::ProcessEntity

The **ISACTInterface::ProcessEntity** method resumes processing of a previously suspended Entity. All Actions that reference a variable that has been modified since the Entity was suspended are evaluated using the new variable state.

```
ISACTRETURN ProcessEntity(
        IHANDLE iHandle
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle

Remarks

All Entities can be processed in one call using ProcessAllEntities.

See Also

CreateEntity, SuspendEntity, ActivateEntity

ISACTInterface::SuspendAllEntities

The **ISACTInterface::SuspendAllEntities** method disables Action processing on all Entities created from Sound Entities from the given Content Bank. Subsequent calls to set variable states store the new states, but no Actions are evaluated. The SuspendAllEntities method can be used as an optimization when the application needs to set multiple variable states without processing Actions after each call. After the application has set all the variables to their new states, all Entities (created from the Content Bank) can be resumed by calling ProcessAllEntities which evaluates all the Actions that reference a variable that has changed state since that Entity was suspended.

```
ISACTRETURN SuspendAllEntities(
        IHANDLE iHandleContentBank
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the Entities to suspend.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank handle

Remarks

SuspendAllEntities is equivalent to calling SuspendEntity on each Entity that was created from a Sound Entity contained in the Content Bank.

See Also

ProcessAllEntities, SuspendEntity

ISACTInterface::ProcessAllEntities

The **ISACTInterface::ProcessAllEntities** method resumes processing of all previously suspended Entities created from the given Content Bank. All Actions that reference a variable that has changed state since the Entity was suspended are evaluated.

```
ISACTRETURN ProcessAllEntities(
        IHANDLE iHandleContentBank
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the Entities to process.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank handle

Remarks

ProcessAllEntities is equivalent to calling ProcessEntity on each Entity created that was created from a Sound Entity contained in the Content Bank.

See Also

SuspendAllEntities, ProcessEntity

ISACTInterface::SetEntityLocalVarState (String based)

The **ISACTInterface::SetEntityLocalVarState** method is used to set the state of the given local variable. Any Actions in the Entity that reference this variable are evaluated using the new variable state (unless the Entity is suspended or in-active).

```
ISACTRETURN SetEntityLocalVarState(
        IHANDLE iHandle,
        const char *szVarName,
        const char *szStateName
);
```

```
ISACTRETURN SetEntityLocalVarState(
        IHANDLE iHandle,
        const unsigned short *szVarName,
        const unsigned short *szStateName
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

szVarName
The name of the variable to change.

szStateName
The name of the new state for the variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	Unknown Variable or State name

Remarks

This method utilizes strings to identify the local variable and state, whereas the SetEntityLocalVarState (Index based) method uses indices to reference the variable and state.

See Also

CreateEntity, SetEntityLocalVarState (Index based), SetEntityGlobalVarState (String based), GetEntityLocalVarState (String based)

ISACTInterface::SetEntityGlobalVarState (String based)

The **ISACTInterface::SetEntityGlobalVarState** method is used to set the state of a global variable in the specified Content Bank. All the Actions in all the Entities created from a Sound Entity from the Content Bank, which reference this variable, are evaluated (unless they are suspended or in-active).

```
ISACTRETURN SetEntityGlobalVarState(
        IHANDLE iHandleContentBank,
        const char *szVarName,
        const char *szStateName
);
```

```
ISACTRETURN SetEntityGlobalVarState(
        IHANDLE iHandleContentBank,
        const unsigned short *szVarName,
        const unsigned short *szStateName
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the global variable to set or the predefined value ISACT_SEARCH_CONTENT_BANKS.

szVarName
The name of the variable to change.

szStateName
The name of the new state for the variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle
ISACT_BAD_VALUE	Unknown Variable or State name

Remarks

This method utilizes strings to identify the variable and state, whereas the SetEntityGlobalVarState (Index based) method uses indices to reference the variable and state.

If the ISACT_SEARCH_CONTENT_BANKS flag is used for the content bank handle, all currently loaded content banks will be checked for the global variable and state and changed appropriately. A value of ISACT_OK will always be returned when using ISACT_SEARCH_CONTENT_BANKS even if no content banks contain the variable and state.

See Also

CreateEntity, SetEntityGlobalVarState (Index based), SetEntityLocalVarState (String based), GetEntityGlobalVarState (String based)

ISACTInterface::SetEntityLocalVarState (Index based)

The **ISACTInterface::SetEntityLocalVarState** method is used to set the state of the given variable. Any Actions in the Entity that reference this variable are evaluated using the new variable state (unless the Entity is suspended or in-active).

```
ISACTRETURN SetEntityLocalVarState(
        IHANDLE iHandle,
        unsigned long ulVariableIndex,
        unsigned long ulStateIndex
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity or the predefined value ISACT_SEARCH_CONTENT_BANKS.

ulVariableIndex
The index of the local variable to change.

ulStateIndex
The index of the new state for the variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle
ISACT_BAD_VALUE	Bad variable or state index

Remarks

If the application wishes to use strings to identify variables and states, then it can use the SetEntityLocalVarState (String based) method.

If the ISACT_SEARCH_CONTENT_BANKS flag is used for the content bank handle, all currently loaded content banks will be checked for the global variable and state index and changed appropriately. A value of ISACT_OK will always be returned when using ISACT_SEARCH_CONTENT_BANKS even if no content banks contain the variable and state.

See Also

CreateEntity, SetEntityLocalVarState (String based), SetEntityGlobalVarState (Index based), GetEntityLocalVarState (Index based)

ISACTInterface::SetEntityGlobalVarState (Index based)

The **ISACTInterface::SetEntityGlobalVarState** method is used to set the state of a global variable contained in the specified Content Bank. All the Actions in all the Entities created from Sound Entities from the Content Bank, which reference this variable are evaluated (unless they are suspended or in-active).

```
ISACTRETURN SetEntityGlobalVarState(
        IHANDLE iHandleContentBank,
        unsigned long ulVariableIndex,
        unsigned long ulStateIndex
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the global variable to set.

ulVariableIndex
The index of the global variable to change.

ulStateIndex
The index of the new state for the variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank handle
ISACT_BAD_VALUE	Bad variable or state index

Remarks

If the application wishes to use strings to identify variables and states, then it can use the SetEntityGlobalVarState (String based) method.

See Also

CreateEntity, SetEntityGlobalVarState (String based), SetEntityLocalVarState (Index based), GetEntityGlobalVarState (Index based)

ISACTInterface::SetEntityMasterRPCLevel

The **ISACTInterface::SetEntityMasterRPCLevel** method sets all the Real-Time Parameter Controllers attached to any of the Players contained in the Entity to the given level. A single RPC level can be controlled using SetEntityRPCLevel.

```
ISACTRETURN SetEntityMasterRPCLevel(
        IHANDLE iHandle,
        float flLevel
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

flLevel
The RPC Level to apply to all the attached RPC's.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

The Master RPC level should be in the range 0 to 1.

One possible use for the Master RPC level is to implement a cross-fade between 2 or more Players in the Entity. This can be achieved by creating two RPC's, one with volume fading out, and one with volume fading in. All Players in the Entity that have the "fade-out" RPC attached, will decrease in volume as the MasterRPC level is increased, while all the Players with the "Fade-In" RPC attached, will increase in volume.

See Also

CreateEntity, SetEntityRPCLevel, GetEntityMasterRPCLevel

ISACTInterface::SetEntityRPCLevel

The **ISACTInterface::SetEntityRPCLevel** is used to set the level of a particular Real-Time Parameter Controller that may be attached to one or more or the Players contained in the Entity.

```
ISACTRETURN SetEntityRPCLevel(
        IHANDLE iHandle,
        const char *szRPCName,
        float flLevel
);
```

```
ISACTRETURN SetEntityRPCLevel(
        IHANDLE iHandle,
        const unsigned short *szRPCName,
        float flLevel
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

szRPCName
Name of the RPC to apply the RPC level to.

flLevel
The new RPC level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle.

Remarks

The RPC level should be between 0 and 1.

If the RPC specified is not attached to any of the Players in the Entity, then the function call simply returns.

To set all RPC's attached to Players in the Entity to the same value, use the SetEntityMasterRPCLevel method.

See Also

CreateEntity, SetEntityMasterRPCLevel, GetEntityRPCLevel

ISACTInterface::SetEntityMode

The **ISACTInterface::SetEntityMode** method is used to indicate if the Entity's Players should be processed as Head Relative. The actual Head Relative mode used by each of the Entity's Players is determined by OR'ing the Entity Mode with the Player's mode (which can be changed as a result of running the Entity's state machine).

```
ISACTRETURN SetEntityMode(
        IHANDLE iHandle,
        unsigned long ulMode
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

ulMode
Head Relative mode (true or false).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityMode

ISACTInterface::SetEntityPosition

The **ISACTInterface::SetEntityPosition** method is used to set the position of the Entity. The positions and orientations of the individual Players within the Entity will be offset by the Entity's position and orientation.

```
ISACTRETURN SetEntityPosition(
        IHANDLE iHandle,
        ISACTVector &vPosition
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

vPosition
Entity's new position.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityPosition

ISACTInterface::SetEntityOrientation (Vector based)

The **ISACTInterface::SetEntityOrientation** method is used to set the vector based orientation of the Entity. The positions and orientations of the individual Players within the Entity will be offset by the Entity's position and orientation. If the application wishes to update the orientation using Euler angles, it can use SetEntityOrientation (Euler based).

```
ISACTRETURN SetEntityOrientation(
        IHANDLE iHandle,
        ISACTVector &vFront,
        ISACTVector &vUp
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

vFront
Entity's new 'forward' direction.

vUp
Entity's new 'up' direction.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityOrientation (Vector based)

ISACTInterface::SetEntityOrientation (Euler based)

The **ISACTInterface::SetEntityOrientation** method is used to set the Euler angle based orientation of the Entity. The positions and orientations of the individual Players within the Entity will be offset by the Entity's position and orientation. If the application wishes to update the orientation using two vectors, it can use SetEntityOrientation (Vector based).

```
ISACTRETURN SetEntityOrientation(
        IHANDLE iHandle,
        float flAzimuth,
        float flHeight,
        float flRoll
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

flAzimuth
Entity's new azimuth angle (between 0 and 360 degrees).

flHeight
Entity's new height angle (between -90 and 90).

flRoll
Entity's new roll angle (between -180 and 180).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityOrientation (Euler based)

ISACTInterface::SetEntityVelocity

The **ISACTInterface::SetEntityVelocity** method is used to set the velocity of the Entity. Each Player in the Entity will have its velocity combined with the Entity's velocity.

```
ISACTRETURN SetEntityVelocity(
        IHANDLE iHandle,
        ISACTVector &vVelocity
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

vVelocity
Entity's new velocity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityVelocity

ISACTInterface::SetEntityPitch

The **ISACTInterface::SetEntityPitch** method is used to set the pitch of the Entity. Each Player in the Entity will have its pitch level multiplied by the Entity's pitch.

```
ISACTRETURN SetEntityPitch(
        IHANDLE iHandle,
        float flPitch
);
```

Parameters

iHandle

Valid Entity handle returned from CreateEntity.

flPitch

Entity's new pitch (value must be >0 and <= 2.0).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityPitch

ISACTInterface::SetEntityGain

The **ISACTInterface::SetEntityGain** method is used to set the gain of the Entity. Each Player in the Entity will have its gain multiplied by the Entity's new gain.

```
ISACTRETURN SetEntityGain(
        IHANDLE iHandle,
        float flGain
);
```

Parameters

iHandle

Valid Entity handle returned from CreateEntity.

flGain

Entity's new gain between 0 (silent) and 1 (full volume).

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity handle

Remarks

None.

See Also

CreateEntity, GetEntityGain

ISACTInterface::GetEntityLocalVarState (String based)

The **ISACTInterface::GetEntityLocalVarState** method is used to retrieve the state of the given local variable.

```
ISACTRETURN GetEntityLocalVarState(
        IHANDLE iHandle,
        const char *szVarName,
        unsigned long *pulStateIndex
);
```

```
ISACTRETURN GetEntityLocalVarState(
        IHANDLE iHandle,
        const unsigned short *szVarName,
        unsigned long *pulStateIndex
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

szVarName
The name of the variable to query the state of.

pulStateIndex
Address of the variable that receives the state index of the given variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	Unknown variable or bad pointer

Remarks

This method utilizes a string to identify the variable, whereas the GetEntityLocalVarState (Index based) method uses indices to reference the variable.

See Also

CreateEntity, GetEntityLocalVarState (Index based), GetEntityGlobalVarState (String based), SetEntityLocalVarState (String based)

ISACTInterface::GetEntityGlobalVarState (String based)

The **ISACTInterface::GetEntityGlobalVarState** method is used to retrieve the state of a global variable contained in the specified Content Bank.

```
ISACTRETURN GetEntityGlobalVarState(
        IHANDLE iHandleContentBank,
        const char *szVarName,
        unsigned long *pulStateIndex
);
```

```
ISACTRETURN GetEntityGlobalVarState(
        IHANDLE iHandleContentBank,
        const unsigned short *szVarName,
        unsigned long *pulStateIndex
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the global variable to query.

szVarName
The name of the variable to query the state of.

pulStateIndex
Address of the variable that receives the state index of the given variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle
ISACT_BAD_VALUE	Unknown variable or bad pointer

Remarks

This method utilizes a string to identify the variable, whereas the GetEntityGlobalVarState (Index based) method uses indices to reference the variable.

See Also

CreateEntity, GetEntityGlobalVarState (Index based), GetEntityLocalVarState (String based), SetEntityGlobalVarState (String based)

ISACTInterface::GetEntityLocalVarState (Index based)

The **ISACTInterface::GetEntityLocalVarState** method is used to retrieve the state of the given local variable.

```
ISACTRETURN GetEntityLocalVarState(
        IHANDLE iHandle,
        unsigned long ulVarIndex,
        unsigned long *pulStateIndex
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

ulVarIndex
The index of the local variable to query.

pulStateIndex
Address of a variable that receives the state index of the given variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle.
ISACT_BAD_VALUE	Unknown variable index or bad pointer

Remarks

If the application wishes to use strings to identify a variable, then it can use the GetEntityLocalVarState (String based) method.

See Also

CreateEntity, GetEntityLocalVarState (String based), GetEntityGlobalVarState (Index based), SetEntityLocalVarState (Index based)

ISACTInterface::GetEntityGlobalVarState (Index based)

The **ISACTInterface::GetEntityGlobalVarState** method is used to retrieve the state of a global variable contained in the specified Content Bank.

```
ISACTRETURN GetEntityGlobalVarState(
        IHANDLE iHandleContentBank,
        unsigned long ulVarIndex,
        unsigned long *pulStateIndex
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the global variable to query.

ulVarIndex
The index of the global variable to query.

pulStateIndex
Address of a variable that receives the state index of the given variable.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle.
ISACT_BAD_VALUE	Unknown variable index or bad pointer

Remarks

If the application wishes to use strings to identify a variable, then it can use the GetEntityGlobalVarState (String based) method.

See Also

CreateEntity, GetEntityGlobalVarState (String based), GetEntityLocalVarState (Index based), SetEntityGlobalVarState (Index based)

ISACTInterface::GetEntityMasterRPCLevel

The **ISACTInterface::GetEntityMasterRPCLevel** method retrieves the master RPC level as set by a previous call to SetEntityMasterRPCLevel. To retrieve the level of a specific RPC an application should use GetEntityRPCLevel.

```
ISACTRETURN GetEntityMasterRPCLevel(
        IHANDLE iHandle,
        float *pflLevel
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pflLevel
The address of a variable that will receive the Master RPC level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle.
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityMasterRPCLevel, GetEntityRPCLevel

ISACTInterface::GetEntityRPCLevel

The **ISACTInterface::GetEntityRPCLevel** is used to retrieve the level of a particular Real-Time Parameter Controller that may be attached to one or more or the Players contained in the Entity. To retrieve the level of the Master RPC an application can use GetEntityMasterRPCLevel.

```
ISACTRETURN GetEntityRPCLevel(
        IHANDLE iHandle,
        const char *szRPCName,
        float *pflLevel
);
```

```
ISACTRETURN GetEntityRPCLevel(
        IHANDLE iHandle,
        const unsigned short *szRPCName,
        float *pflLevel
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

szRPCName
Name of the RPC to query the level of.

pflLevel
Address of a floating point variable that receives the level of the given RPC.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle.
ISACT_BAD_VALUE	Unknown or unused RPC, or bad pointer

Remarks

None.

See Also

CreateEntity, GetEntityMasterRPCLevel, SetEntityRPCLevel

ISACTInterface::GetEntityMode

The **ISACTInterface::GetEntityMode** method is used to retrieve the head-relative mode of the Entity, as set by a previous call to SetEntityMode.

```
ISACTRETURN GetEntityMode(
        IHANDLE iHandle,
        unsigned long *pulMode
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pulMode
Address of an unsigned long that receives the Head Relative mode of the Entity (true or false)

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityMode

ISACTInterface::GetEntityPosition

The **ISACTInterface::GetEntityPosition** method is used to retrieve the position of the Entity as set by a previous call to SetEntityPosition.

```
ISACTRETURN GetEntityPosition(
        IHANDLE iHandle,
        ISACTVector *pvPosition
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pvPosition
Address of an ISACTVector that will receive the Entity's position.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityPosition

ISACTInterface::GetEntityOrientation (Vector based)

The **ISACTInterface::GetEntityOrientation** method is used to retrieve the vector based orientation of the Entity as set by a previous call to SetEntityOrientation (Vector based).

```
ISACTRETURN GetEntityOrientation(
        IHANDLE iHandle,
        ISACTVector *pvFront,
        ISACTVector *pvUp
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pvFront
Address of an ISACTVector that will receive the Entity's 'forward' direction.

vUp
Address of an ISACTVector that will receive the Entity's 'up' direction.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityOrientation (Vector based)

ISACTInterface::GetEntityOrientation (Euler based)

The **ISACTInterface::GetEntityOrientation** method is used to retrieve the Euler angle based orientation of the Entity as set by a previous call to SetEntityOrientation (Vector based).

```
ISACTRETURN GetEntityOrientation(
        IHANDLE iHandle,
        float *pflAzimuth,
        float *pflHeight,
        float *pflRoll
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pflAzimuth
Address of a floating point variable that will receive the Entity's azimuth angle.

pflHeight
Address of a floating point variable that will receive the Entity's height angle.

pflRoll
Address of a floating point variable that will receive the Entity's roll angle.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityOrientation (Euler based)

ISACTInterface::GetEntityVelocity

The **ISACTInterface::GetEntityVelocity** method is used to retrieve the velocity of the Entity, as set by a previous call to SetEntityVelocity.

```
ISACTRETURN GetEntityVelocity(
        IHANDLE iHandle,
        ISACTVector *pvVelocity
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pvVelocity
Address of an ISACTVector that will receive the Entity's velocity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle.
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityVelocity

ISACTInterface::GetEntityPitch

The **ISACTInterface::GetEntityPitch** method is used to retrieve the pitch of the Entity, as set by a previous call to SetEntityPitch.

```
ISACTRETURN GetEntityPitch(
        IHANDLE iHandle,
        float *pflPitch
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pflPitch
Address of a floating point variable that receives the Entity's pitch level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityPitch

ISACTInterface::GetEntityGain

The **ISACTInterface::GetEntityGain** method is used to retrieve the gain of the Entity, as set by a previous call to SetEntityGain.

```
ISACTRETURN GetEntityGain(
        IHANDLE iHandle,
        float *pflGain
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pflGain
Address of a floating point variable that receives the Entity's gain level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

None.

See Also

CreateEntity, SetEntityGain

ISACTInterface::GetEntityMaxGain

The **ISACTInterface::GetEntityMaxGain** method is used to retrieve the loudest gain that any piece of Content referenced by the Entity Script would have at a given 3D position.

```
ISACTRETURN GetEntityMaxGain(
        IHANDLE iHandle,
        ISACTVector *pvPosition,
        unsigned long ulHeadRelativeMode,
        float *pflGain
);
```

Parameters

iHandle

Valid Entity handle returned from CreateEntity.

pvPosition

Address of an ISACTVector containing the position for the Entity that will be used to calculate the attenuation over distance. If this is NULL then the current position of the Entity will be used to calculate attenuation gain.

ulHeadRelativeMode

If true (set to 1) then the position pointed to by pvPosition will be treated as relative to the Listener's position, rather than an absolute co-ordinate. If pvPosition is NULL, then the Entity's Head Relative mode setting is used instead.

pflGain

Address of a floating point variable that receives the Entity's Max gain level.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The Max Gain is calculated by finding the loudest maximum gain of each piece of Content that could be played by the Entity (not necessarily what *is* currently playing). The MaxGain calculation takes into account distance attenuation (if any), and content gain (content level and sub-mix level). It does not include gain adjustments that are part of the content itself (e.g volume randomization in a Sound Event, or gain levels in a Sequence).

Note – to calculate the maximum gain of an Entity ignoring 3D attenuation, pass in a vector of {0, 0, 0} and set ulHeadRelativeMode equal to 1.

See Also

CreateEntity, GetContentMaxGain, GetAudioPlayerMaxGain

ISACTInterface::GetNumLocalVars

The **ISACTInterface::GetNumLocalVars** method is used to retrieve the number of local variables defined in the given Entity.

```
ISACTRETURN GetNumLocalVars(
        IHANDLE iHandle,
        unsigned long *pulNumVars
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

pulNumVars
Address of an unsigned long variable that receives the number of local variables in the Entity.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The number of Global variables can be determined using GetNumGlobalVars.

See Also

CreateEntity, GetNumGlobalVars

ISACTInterface::GetNumGlobalVars

The **ISACTInterface::GetNumGlobalVars** method is used to retrieve the number of global variables contained in the given Content Bank.

```
ISACTRETURN GetNumGlobalVars(
        IHANDLE iHandleContentBank,
        unsigned long *pulNumVars
);
```

Parameters

iHandleContentBank
Handle of Content Bank to query.

pulNumVars
Address of an unsigned long variable that receives the number of global variables in the Content Bank.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The number of local variables can be determined using GetNumLocalVars.

See Also

GetNumLocalVars

ISACTInterface::GetNumLocalVarStates

The **ISACTInterface::GetNumLocalVarStates** method is used to retrieve the number of states defined in the given local variable of the Entity.

```
ISACTRETURN GetNumLocalVarStates(
        IHANDLE iHandle,
        unsigned long ulVarIndex,
        unsigned long *pulNumStates
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

ulVarIndex
Index of variable to query.

pulNumStates
Address of an unsigned long variable that receives the number of states.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The number of states in a global variable can be determined using GetNumGlobalVarStates.

See Also

CreateEntity, GetNumGlobalVarStates

ISACTInterface::GetNumGlobalVarStates

The **ISACTInterface::GetNumGlobalVarStates** method is used to retrieve the number of states defined in a global variable contained in the specified Content Bank.

```
ISACTRETURN GetNumGlobalVarStates(
        IHANDLE iHandleContentBank,
        unsigned long ulVarIndex,
        unsigned long *pulNumStates
);
```

Parameters

iHandleContentBank
Handle of Content Bank containing the global variable to query.

ulVarIndex
Index of global variable to query.

pulNumStates
Address of an unsigned long variable that receives the number of states.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The number of states in a local variable can be determined using GetNumLocalVarStates.

See Also

GetNumLocalVarStates

ISACTInterface::GetLocalVarName

The **ISACTInterface::GetLocalVarName** method is used to retrieve the name of the given variable index of the Entity.

```
ISACTRETURN GetLocalVarName(
        IHANDLE iHandle,
        unsigned long ulVarIndex,
        char *pszName,
        unsigned long ulNameLength
);
```

```
ISACTRETURN GetLocalVarName(
        IHANDLE iHandle,
        unsigned long ulVarIndex,
        unsigned short *pszName,
        unsigned long ulNameLength
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

ulVarIndex
Index of the variable to find the name of.

pszName
Address of a string that will receive the name of the variable.

ulNameLength
Size of the string pointed to by pszName.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The name of a Global variable can be determined using GetGlobalVarName.

See Also

GetGlobalVarName

ISACTInterface::GetGlobalVarName

The **ISACTInterface::GetGlobalVarName** method is used to retrieve the name of a global variable contained in the specified Content Bank.

```
ISACTRETURN GetGlobalVarName(
        IHANDLE iHandleContentBank,
        unsigned long ulVarIndex,
        char *pszName,
        unsigned long ulNameLength
);
```

```
ISACTRETURN GetGlobalVarName(
        IHANDLE iHandleContentBank,
        unsigned long ulVarIndex,
        unsigned short *pszName,
        unsigned long ulNameLength
);
```

Parameters

iHandleContentBank
Handle of Content Bank containing the global variable to query.

ulVarIndex
Index of the variable to retrieve the name of.

pszName
Address of a string that will receive the name of the variable.

ulNameLength
Size of the string pointed to by pszName.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The name of a local variable can be determined using GetLocalVarName.

See Also

GetLocalVarName

ISACTInterface::GetLocalVarStateName

The **ISACTInterface::GetLocalVarStateName** method is used to retrieve the name of the given state index of the specified variable of the Entity.

```
ISACTRETURN GetLocalVarStateName(
        IHANDLE iHandle,
        unsigned long ulVarIndex,
        unsigned long ulStateIndex,
        char *pszName,
        unsigned long ulNameLength
);
```

```
ISACTRETURN GetLocalVarStateName(
        IHANDLE iHandle,
        unsigned long ulVarIndex,
        unsigned long ulStateIndex,
        unsigned short *pszName,
        unsigned long ulNameLength
);
```

Parameters

iHandle
Valid Entity handle returned from CreateEntity.

ulVarIndex
Index of the variable to query.

ulStateIndex
Index of the state to find the name of.

pszName
Address of a string that will receive the name of the variable.

ulNameLength
Size of the string pointed to by pszName.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Entity Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The name of a Global variable state can be determined using GetGlobalVarStateName.

See Also

GetGlobalVarStateName

ISACTInterface::GetGlobalVarStateName

The **ISACTInterface::GetGlobalVarStateName** method is used to retrieve the name of a state of a global variable contained in the specified Content Bank.

```
ISACTRETURN GetGlobalVarStateName(
        IHANDLE iHandleContentBank,
        unsigned long ulVarIndex,
        unsigned long ulStateIndex,
        char *pszName,
        unsigned long ulNameLength
);
```

```
ISACTRETURN GetGlobalVarStateName(
        IHANDLE iHandleContentBank,
        unsigned long ulVarIndex,
        unsigned long ulStateIndex,
        unsigned short *pszName,
        unsigned long ulNameLength
);
```

Parameters

iHandleContentBank
Handle of the Content Bank containing the global variable to query.

ulVarIndex
Index of the variable to query.

ulStateIndex
Index of the state to find the name of.

pszName
Address of a string that will receive the name of the variable.

ulNameLength
Size of the string pointed to by pszName.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error.
ISACT_BAD_HANDLE	Specified handle is not a valid Content Bank Handle
ISACT_BAD_VALUE	A parameter is invalid or out of range

Remarks

The name of a Local variable state can be determined using GetLocalVarStateName.

See Also

GetLocalVarStateName

Audio Group Functions

Handle manipulation	GetGroupHandle
	IsValidGroupHandle
Group	SetGroupGain
	GetGroupGain

ISACTInterface::GetGroupHandle

The **ISACTInterface::GetGroupHandle** method is used to obtain a handle to a Group. Any playable Content (including Samples) can be assigned to a particular Group. When Content and Samples are loaded, they are automatically assigned to the appropriate group as determined by the Sound Designer using the ISACT Production Studio.

```
ISACTRETURN GetGroupHandle(
        IHANDLE *piHandle,
        const char *szGroupName
);
```

```
ISACTRETURN GetGroupHandle(
        IHANDLE *piHandle,
        const unsigned short*szGroupName
);
```

Parameters

piHandle
Address of an IHANDLE that will receive the handle of the Group.

szGroupName
Name of the requested Group.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is invalid or out of range
ISACT_NOT_FOUND	Group name was not found
ISACT_NOT_INITIALIZED	ISACT Interface has not been initialized

Remarks

A Group Handle is used to set or get the volume of particular Group. Group volumes are combined with other volume modifiers such as Real-Time Parameter Controllers.

See Also

IsValidGroupHandle, SetGroupGain, GetGroupGain

ISACTInterface::IsValidGroupHandle

The **ISACTInterface::IsValidGroupHandle** method can be used to verify if a handle is a Group handle.

```
ISACTRETURN IsValidGroupHandle (
        IHANDLE iHandle
);
```

Parameters

iHandle
IHANDLE data type to be verified.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_HANDLE	The specified handle is not a valid Group handle

Remarks

If the passed in handle is a valid Group handle the return identifier is ISACT_OK. If it is not a valid handle the return identifier is ISACT_BAD_HANDLE.

See Also

GetGroupHandle

ISACTInterface::SetGroupGain

The **ISACTInterface::SetGroupGain** method can be used to set the volume on all Content and Samples assigned to the Group.

```
ISACTRETURN SetGroupGain (
        IHANDLE iHandle,
        float flGain,
        unsigned long ulFadeTime = 0
);
```

Parameters

iHandle
Valid Group handle.

flGain
Group gain between 0 and 1

ulFadeTime
Duration to fade to new level in milliseconds.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is out of range
ISACT_BAD_HANDLE	The specified handle is not a valid Group handle

Remarks

All Content and Samples will use the new gain. If a fade-time is specified, the group volume will be ramped to the level over ulFadeTime milliseconds.

See Also

GetGroupGain

ISACTInterface::GetGroupGain

The **ISACTInterface::GetGroupGain** method is used to retrieve the Group volume.

```
ISACTRETURN GetGroupGain (
        IHANDLE iHandle,
        float *pflGain
);
```

Parameters

iHandle
Valid Group handle.

**pflGain*
Address of floating point variable to receive gain.

Return Values

Identifier	Description
ISACT_OK	Method succeeded without error
ISACT_BAD_VALUE	A parameter is out of range
ISACT_BAD_HANDLE	The specified handle is not a valid Group handle

Remarks

None.

See Also

SetGroupGain

Data Types

PLAYER_STATUS

```
enum PLAYER_STATUS {
        PLAYER_STOPPED,
        PLAYER_RELEASE,
        PLAYER_PAUSED,
        PLAYER_PLAYING,
        PLAYER_PENDING_TRANSITION,
        PLAYER_TRANSITIONING
};
```

The PLAYER_STATUS values are used with the PlayerStatus member of the SPlayerStatus structure. These values are returned by the ISACT engine from the GetAudioPlayerStatus function.

PLAYER_STOPPED
The player is stopped and not playing any content.

PLAYER_RELEASE
The player was stopped but the content is still playing a release section of audio.

PLAYER_PAUSED
The player is paused and not playing any content.

PLAYER_PLAYING
The player is playing the primary content.

PLAYER_PENDING_TRANSITION
The player was told to transition but is waiting for the designated start position to be reached.

PLAYER_TRANSITIONING
The player is transitioning the primary content to the transition content..

TRANSITION_START

```
enum TRANSITION_START {
        START_IMMEDIATE,
        START_MARKER,
        START_AFTER,
        START_AFTER_LOOP
};
```

The TRANSITION_START enumerations are used with the StartType member of the STransition structure. This structre is used with the TransitionAudioPlayer functions. These values identify when a transition should start.

START_IMMEDIATE

Transition should start immediately.

START_MARKER
Transition should start at the next marker in the primary content.

START_AFTER
Transition should start after the primary content is finished. Note if the primary content is set to loop it must complete all loops before the content will transition.

START_AFTER_LOOP
Transition should start at the next loop point of the primary content, or at the end of the primary content if no loop value was set.

Structures

STransition

```
struct STransition {
        TRANSITION_START StartType;
        float    fTransitionLength,
                 fPrimaryEndTime,
                 fPrimaryEndLevel,
                 fTransitionStartTime,
                 fTransitionStartLevel;
};
```

Members

StartType
Valid TRANSITION_START enumeration value.

fTransitionLength
Total length of the transition in seconds.

fPrimaryEndTime
Time offset in seconds to end primary content. Must be less than or equal to fTransitionLength.

fPrimaryEndLevel
Volume level to decay primary content to over time length of fPrimaryEndTime. Used as a multiplier this value needs to be between 0 and 1 inclusive.

fTransitionStartTime
Time offset in seconds to start transition content. Must be less than or equal to fTransitionLength.

SPlayerStatus

```
struct SPlayerStatus {
  PLAYER_STATUS    PlayerStatus;
  lHANDLE          iHandePrimaryContent,
                   iHandlePrimaryMarker,
                   iHandleTransitionContent,
                   iHandleTransitionMarker;
  long             lPrimaryLoopCount,
                   lTransitionLoopCount;
  unsigned long    ulMilliseconds;
};
```

Members

PlayerStatus
Current status of the player (see PLAYER_STATUS).

iHandlePrimaryContent
Handle to primary content attached to the player. Null if no primary content is attached.

iHandlePrimaryMarker
Marker handle of starting point in primary content. Null if no marker was passed in.

iHandleTransitionContent
Handle to transition content attached to the player. Null if no transition content is attached.

iHandleTrnasitionMarker
Marker handle of starting point in transition content. Null if no marker was passed in.

lPrimaryLoopCount
Loop count passed in when player was played.

lTransitionLoopCount
Loop count passed in when player was transitioned.

ulMilliseconds
How long player has been playing in milliseconds.

SPlayerMetricTime

```
struct SPlayerMetricTime {
        unsigned long   ulMeasures,
                        ulBeats,
                        ulClocks,
                        ulTimeSignature,
                        ulTempo,
                        ulTotalClocks,
                        ulStatusFlags;
};
```

Members

ulMeasures
Number of measures.

ulBeats
Number of beats.

ulClocks
Number of clocks.

ulTimeSignature
Time signature.

ulTempo
Tempo.

ulTotalClocks
Total clocks.

ulStatusFlags
Status flags.

SPlayerRealTime

```
struct SPlayerRealTime {
        unsigned long   ulHours,
                        ulMinutes,
                        ulSeconds,
                        ulFrames,
                        ulClocks,
                        ulTotalClocks,
                        ulStatusFlags;
};
```

Members

ulHours
Number of hours.

ulMinutes
Number of minutes.

ulSeconds
Number of seconds.

ulFrames
Number of frames.

ulClocks
Clocks.

ulTotalClocks
Total clocks.

ulStatusFlags
Status flags.

SContentInfo

```
struct SContentInfo {
        unsigned long   ulMinDuration,
                        ulMaxDuration;
};
```

Members

ulMinDuration
Minimal duration of playback in milliseconds.

ulMaxDuration
Maximal duration of playback in milliseconds.

ISACTDeviceCaps

```
#define EAX2SUPPORT                 0x01
#define EAX3SUPPORT                 0x02
#define EAX4SUPPORT                 0x04

struct ISACTDeviceCaps
{
        char            szDeviceName[256];
        unsigned long   ulMaxVoices;
        unsigned long   ulEAXSupported;
        unsigned long   ulEAXEmulated;
};
```

Members

szDeviceName
Open AL Device name.

ulMaxVoices
Maximum number of Open AL Sources that can be created.

ulEAXSupported
Versions of EAX natively supported (bit flags).

ulEAXEmulated
Versions of EAX that are emulated (bit flags).

SStreamFunctions

```
typedef struct
{
        void * (*OpenStream) (const char *szFilename, const char *szMode);
        int    (*CloseStream) (void *pStream);
        size_t (*ReadStream) (void *pBuffer, size_t size, size_t count, void *pStream);
        int    (*SeekStream) (void *pStream, long lOffset, long lOrigin);
        long   (*TellStream) (void *pStream);
} SStreamFunctions;
```

Members

OpenStream
Pointer to file open function (parameters equivalent to fopen)

CloseStream
Pointer to file close function (passes in handle returned from OpenStream)

ReadStream
Pointer to file read function (parameters equivalent to fread)

SeekStream
: Pointer to file seek function (parameters equivalent to fseek)

TellStream
: Pointer to file tell function (parameters equivalent to ftell)

Macros

FORCE_SAMPLE_BANK

```
#define FORCE_SAMPLE_BANK(a) (a|0x80000)
```

The FORCE_SAMPLE_BANK macro can be used with the SetContentBankSampleBankOffset function to force a content bank to use a specific sample bank index. As an example you could force a content bank to use sample bank 7 for all its samples as follows.

```
ISACT->SetContentBankSampleBankOffset(hContentBank FORCE_SAMPLE_BANK(7));
```

See Also

SetContenBankSampleBankOffset