Creative labs'

*EAX*Manager .DLL

reference documentation

Techincal : kcharley@creativelabs.com
Documentation : cvogelsang@creativelabs.com

What is EAXManager ?

EAXManager is a COM interface that resides in a DLL that will process and handle `.EAL` files created by EAGLE™. EAGLE™ is a tool which enables the complete sounddesign to be made *and* tested by the sounddesigner without having to revert to a programmer. EAGLE™ will then export all this information to a single file, which can then be read by EAXManager and queried for information at runtime. All the application has to do is load the file into EAXManager, query it for the static properties, such as listener distance units and so on. Then at runtime the application can query EAXManager with the listener position to get the actual environment the listener is in, it can then query EAXManager with the position of each source to get extended information such as occlusion and obstruction values. Note that EAXManager does NOT take care of setting the actual DirectSound3D parameters, it merely returns the proper values for these fields, the application then has to make sure the actual values are set to DirectSound3D.

In the end EAXManager only takes a day to integrate into your game engine and will take a lot of sound related work away from the programmer, that had to be done by the sound designer in the first place. It gives more creative freedom to the sound designer without loading the programmer with more work.

The rest of this document will show a step by step implementation for EAXManager into *your* application including sample source code fragments. Followed by a EAXManager COM interface function level description.

Adding EAXManager functionality into your application

EAXManager was created so that a Sound Designer could create and test the data needed for advanced 3D audio APIs such as DirectSound and EAX, outside of the programming environment. It is important to understand that EAXManager does not connect with these APIs directly, but is simply the translation layer between the Sound Designer and the Programmer. It is still the Programmers responsibility to send the data retrieved from EAXManager to the Audio API. In this way the application still has an opportunity to alter or ignore the data based on certain conditions that would be unknown to EAXManager. Following are step by step instructions with full code examples on how to implement EAXManager into your application.

Step 1: EAXManager Initialization

Although EAXManager is a COM object, it does not require registration of the CLSID to retrieve the interface. A simple way to initialize the EAXMan.dll is simply to link your application with the EAXMan.lib file and call the EaxManagerCreate function to retrieve the EAXMANAGER interface as illustrated below.

```
#include <eaxman.h>

LPEAXMANAGER            eaxManager = NULL;

//Retrieve the EaxMangerInterface
HRESULT hr = EaxMangerCreate(&eaxManger);

//Check for possible error
if ( hr != EM_OK )
  eaxManager=NULL;
```

If you do not wish to link your application with the EAXMan.lib file, you can link with the EAXMan.dll via the LoadLibrary function. After successfully loading the DLL you will need to retrieve the address of the EaxManagerCreate function using GetProcAddress as follows.

```
#include <eaxman.h>

HINSTANCE               eaxDLL = NULL;
LPEAXMANAGERCREATE      eaxCreate = NULL;
LPEAXMANAGER            eaxManager = NULL;

//Attempt to load the EAXManger.dll
eaxDLL = LoadLibrary("EAXMan.dll");

if ( eaxDLL != NULL )
{
  // Attempt to retrieve the EaxManagerCreate function address.
  eaxCreate = GetProcAddress(eaxDLL, "EaxManagerCreate");

  if ( eaxCreate != NULL )
  {
    // Attempt to create the EAXManager Interface.
    HRESULT hr = eaxCreate(&eaxManager);

    if ( hr != EM_OK )
     eaxManager = NULL;
  }
}
```

You can register EAXMan.dll as a COM object in the registery either from your applications installation, or using an external program such as Regsvr32.exe. Once registered, the EAXMANAGER interface can be created using the CoCreateInstance method.

```
#include <eaxman.h>

LPEAXMANAGER            eaxManager;

//Initialise COM
CoInitialize(NULL);

//Create eaxManager object
if ( CoCreateInstance(CLSID_EAXMANAGER,
                      NULL,
                      CLSCTX_INPROC_SERVER,
                      IID_IEaxManager,
                      &eaxManager)!= S_OK )
  eaxManager=NULL;
```

Step 2: Loading an Environmental Audio Library Dataset

After the creation of the EAXManager interface, it's time to load an environmental dataset (`.eal file`) as exported by EAGLE™.

```
#include <dsound.h>
#include <eaxman.h>

LPDIRECTSOUND3DLISTENER Listener;
LISTENERATTRIBUTES ListenerAttributes;

// Attempt to load the eal file.
HRESUT hr = eaxManager->LoadDataSet("test.eal",0);

if ( hr != EM_OK )
  // Handle error here.
```

Alternatively a file can be loaded from memory by specifying the flag EMFLAG_LOADFROMMEMORY and supplying a pointer to the memory image of the file instead of a file name as follows.

```
#include <dsound.h>
#include <eaxman.h>

LPDIRECTSOUND3DLISTENER Listener;
LISTENERATTRIBUTES ListenerAttributes;

// Here pData is assumed to be a valid memory pointer.
HRESUT hr = eaxManager->LoadDataSet(pData,EMFLAG_LOADFROMMEMORY);

if ( hr != EM_OK )
  // Handle error here.
```

Step 3: Setting the Listeners Attributes

Once the library file has been loaded, the first set of data to be queried should be the Listeners Attributes. These need to be set only once and before any 3D audio is played. The following code fragment is a function which retrieves the Listener attributes from EAXManager, and then sends the data on to DirectSound.

```
#include <dsound.h>
#include <eaxman.h>

BOOL InitListenerAttributes(LPEAXMANGER eaxManager,
                            LPDIRECTSOUND3DLISTENER Listener)
{
  LISTENERATTRIBUTES la;
  HRESULT hr = eaxManager->GetListenerAttributes(&la);

  if ( hr != EM_OK )
    return FALSE;

  Listener->SetDistanceFactor(
            la.fDistanceFactor,DS3D_DEFERRED);
  Listener->SetRolloffFactor(
            la.fRolloffFactor,DS3D_DEFERRED);
  Listener->SetDopplerFactor(
            la.fDopplerFactor,DS3D_DEFERRED);
  Listener->CommitDeferredSettings();

  Return TRUE;
}
```

Step 4: Setting the Source Attributes (Source Preset)

Source attributes are the initialization parameters for each individual sound (referred to as a "Source Preset"). To retrieve the proper settings for a sound, a naming convention will need to be agreed upon by both the Sound Designer and the Programmer. By default the EAGLE™ tool names a source preset after the wave file name minus the .wav extension. However the Sound Designer is free to rename the preset to whatever they wish. Once a sound is loaded and its 3D buffer created, its initialization parameters can be queried for based on the agreed upon naming convention. The following is a routine that will query EAXManager for the sound source properties and then send those parameters to DirectSound and EAX.

```
#include <dsound.h>
#include <eaxman.h>

BOOL InitSourcePreset(LPEAXMANAGER eaxManager,
                      LPDIRECTSOUND3DBUFFER lpds3db,
                      char *szPresetName)
{
  // Retrieve the preset ID number.
  long lPresetID;
  if ( eaxManager->GetSourceID(szPresetName, &lPresetID) != EM_OK )
    return FALSE;

  // Next get the actual preset data.
  SOURCEATTRIBUTES sa;
  if ( eaxManager->GetSourceAttributes(lPresetID, &sa) != EM_OK )
    return FALSE;

  // Set the DSound initialization parameters.
  lpds3db->SetMinDistance(sa.fMinDistance,DS3D_IMMEDIATE);
  lpds3db->SetMaxDistance(sa.fMaxDistance,DS3D_IMMEDIATE);
  lpds3db->SetConeOrientation(sa.fConeXdir,sa.fConeYdir,sa.fConeZdir,DS3D_IMMEDIATE );
  lpds3db->SetConeOutsideVolume(sa.lConeOutsideVolume,DS3D_IMMEDIATE);
  lpds3db->SetConeAngles(sa.ulInsideConeAngle,sa.ulOutsideConeAngle, DS3D_IMMEDIATE);

  // Get a propertyset from the buffer and set the EAX parameters.
  LPKSPROPERTYSET lpksps;

  if ( lpds3db->QueryInterface(IID_IKsPropertySet, (LPVOID *)&lpksps) == DS_OK )
  {
     lpksps->Set(DSPROPSETID_EAX_BufferProperties,
                 DSPROPERTY_EAXBUFFER_ALLPARAMETERS,
                 NULL,
                 0,
                 &sa.eaxAttributes,
                 sizeof(EAXBUFFERPROPERTIES));

    lpksps->Release();
    return TRUE;
  }

  return FALSE;
}
```

Step 5: Setting Environment Attributes (Environment Preset)

The Environment Attributes (known as the Environment Preset) are parameters that apply to the EAX listener to create the reverberated audio environment that the listener is in. The Sound Designer creates Environment Presets with EAGLE™ and gives each one an appropriate name. Again here a naming convention needs to be agreed upon between the Sound Designer and the Programmer. The following coded procedure can then be used to set a new Environment Preset. A DirectSound 3D buffer is needed to retrieve a propertyset interface. Any hardware 3D buffer will do.

```
#include <dsound.h>
#include <eaxman.h>

BOOL SetNewEnvironment(LPEAXMANAGER eaxManager,
                       LPDIRECTSOUND3DBUFFER lpds3db,
                       char *szPresetName)
{
  // Retrieve the preset ID number.
  long lPresetID;
  if ( eaxManager->GetEnvironmentID(szPresetName, &lPresetID) != EM_OK )
    return FALSE;

  // Next get the actual preset data.
  EAXLISTENERPROPERTIES eaxlp;
  if ( eaxManager->GetEnvironmentsAttributes(lPresetID, &eaxlp) != EM_OK )
    return FALSE;

  // Get a propertyset from the buffer.
  LPKSPROPERTYSET lpksps;
  if ( lpds3db->QueryInterface(IID_IKsPropertySet, (LPVOID *)&lpksps) == DS_OK )
  {
     lpksps->Set(DSPROPSETID_EAX_ListenerProperties,
                 DSPROPERTY_EAXLISTENER_ALLPARAMETERS,
                 NULL,
                 0,
                 &eaxlp,
                 sizeof(EAXLISTENERPROPERTIES));

    lpksps->Release();
    return TRUE;
  }

  return FALSE;
}
```

Step 6: Setting Material Attributes (Material Preset)

Material Attributes are applied to 3D sound sources to make them sound as if they were coming from the other side of a wall or being obstructed by some large object. These attributes are created by the Sound Designer within EAGLE™ as a Material Preset. Each preset can be given a unique name. The Programmer can then use this name to retrieve the preset data from EAXManager. There are currently 2 types of Material Presets. Occluding (EMMATERIAL_OCCLUDES) and Obstructing (EMMATERIAL_OBSTRUCTS). The following coded procedure can be used to apply a Material Preset to a sound.

```
#include <dsound.h>
#include <eaxman.h>

BOOL SetSoundMaterial(LPEAXMANAGER eaxManager,
                      LPDIRECTSOUND3DBUFFER lpds3db,
                      char *szPresetName)
{
  // Retrieve the preset ID number.
  long lPresetID;
  if ( eaxManager->GetMaterialID(szPresetName, &lPresetID) != EM_OK )
    return FALSE;

  // Next get the actual preset data.
  MATERIALATTRIBUTES ma;
  if ( eaxManager->GetMaterialAttributes(lPresetID, &ma) != EM_OK )
    return FALSE;

  // Get a propertyset from the buffer.
  LPKSPROPERTYSET lpksps;
  if ( lpds3db->QueryInterface(IID_IKsPropertySet, (LPVOID *)&lpksps) == DS_OK )
  {
    long lValue;
    float fValue;

   switch ( ma.dwFlags )
   {
 case EMMATERIAL_OBSTRUCTS:
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OBSTRUCTION,NULL,0,&ma.lLevel,sizeof(long));
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OBSTRUCTIONLFRATIO,
              NULL,0,&ma.fLFRatio,sizeof(float));

  lValue = EAXBUFFER_DEFAULTOCCLUSION;
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSION,
              NULL,0,&lValue,sizeof(long));
  fValue = EAXBUFFER_DEFAULTOCCLUSIONLFRATIO;
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSIONLFRATIO,
              NULL,0,&fValue,sizeof(float));
  fValue = EAXBUFFER_DEFAULTOCCLUSIONROOMRATIO;
  lpksps->Set(DSPROPSETID_EAX_BufferProperties, DSPROPERTY_EAXBUFFER_OCCLUSIONROOMRATIO,
              NULL,0,&fValue,sizeof(float));
  break;

 case EMMATERIAL_OCCLUDES:
  lValue = EAXBUFFER_DEFAULTOBSTRUCTION;
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OBSTRUCTION,NULL,0,&lLevel,sizeof(long));
  fValue = EAXBUFFER_DEFAULTOBSTRUCTIONLFRATIO;
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OBSTRUCTIONLFRATIO,
              NULL,0,&fValue,sizeof(float));

  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSION,NULL,0,&ma.lLevel,sizeof(long));
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSIONLFRATIO,
              NULL,0,&ma.fLFRatio,sizeof(float));
  lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSIONROOMRATIO,
             NULL,0,&ma.fLFRatio,sizeof(float));
  break;

 default:
  break;
    }

    lpksps->Release();
    return TRUE;
 }

  return FALSE;
}
```

Adding EAXManager geometry functionality to your application

Aside from the ability to retrieve Listener Attribute data, as well as Environment, Source, and Material Preset data, EAXManager provides functionality to determine what data needs to be applied based on 3D position. For instance, you can find out what Environment Preset should be used based on the Listeners position in the applications geometry. Or what Material Preset to apply to a sound based on its 3D position compared to the Listeners position. The following topics “Getting the Listeners Dynamic Attributes” and “Getting the Sounds Dynamic Attributes” are instructions with full code examples on accomplishing these tasks. If you do not yet know how to initialize the EAXMan.dll and/or load the appropriate dataset file, you should first read the section entitled “Adding EAXManager functionality to your application”.

Getting the Listeners Dynamic Attributes

As the Listener moves about in 3D space, he/she will enter many different areas with different reverberating properties. Using Eagles geometry capabilities, the Sound Designer is able to tag these areas with unique Environment Presets they have created. This information can be retrieved from EAXManager by using the Listeners 3D world coordinates. It is important to understand that both EAGLE™ and EAXManager use the same coordinate system as DirectSound with positive Z forward, positive Y up, and positive X to the right. Any coordinates that do not conform to this system must first be converted before they are sent to EAXManager. The following coded procedure shows how to retrieve the proper environment data using the listener position, and send that data to EAX. A DirectSound 3D buffer is needed to retrieve a propertyset interface. Any hardware 3D buffer will do.

```
#include <dsound.h>
#include <eaxman.h>

BOOL SetListenersDynamicAttributes(LPEAXMANAGER eaxManager,
                          LPDIRECTSOUND3DBUFFER lpds3db,
                          float fXpos, float fYpos, float fZpos)
{
  // A static variable is needed to save the last used
  // environment preset ID.
  long lLastID = EMFLAG_IDNONE;

  // An EMPOINT struct is initialized with the listeners
  // position. If the coordinate system being used differed,
  // here is where you would want to convert the data
  // (e.g. swap Y and Z if needed).
  EMPOINT ep;
  ep.fX = fXpos; ep.fY = fYpos; ep.fZ = fZpos;

  long lNewID;
  HRESULT hr = eaxManager->GetListenerDynamicAttributes(0,
                                          &ep,
                                          &lNewID,
                                          EMFLAG_LOCKPOSITION);
  if ( hr != EM_OK )
    return FALSE;

  // Check if this is a new environment.
  if ( lNewID == lLastID )
    return FALSE;
  else
    lLastID = lNewID;

  // Next get the actual preset data.
  EAXLISTENERPROPERTIES eaxlp;
  if ( eaxManager->GetEnvironmentsAttributes(lNewID, &eaxlp) != EM_OK )
    return FALSE;

  // Get a propertyset from the buffer.
  LPKSPROPERTYSET lpksps;
  if ( lpds3db->QueryInterface(IID_IKsPropertySet, (LPVOID *)&lpksps) == DS_OK )
  {
    lpksps->Set(DSPROPSETID_EAX_ListenerProperties,
                DSPROPERTY_EAXLISTENER_ALLPARAMETERS,
                NULL,0,&eaxlp,sizeof(EAXLISTENERPROPERTIES));
    lpksps->Release();
    return TRUE;
  }
  return FALSE;
}
```

Getting the Sounds Dynamic Attributes

As the Listener and/or Sound Source move around in 3D space, the path between the Sound and the Listener can become obscured. Either by a large object or because the Listener is in another room and the Sound is traveling through a wall. Within EAGLE™, the Sound Designer will create many Material Presets that should be applied to a Sound when this occurs. EAXManager can then inform an application as to when to apply Material Attributes to a Sound, and what Attributes to apply. The following is a coded routine that can be used to update a Sound Sources Material Attributes based on its position in 3D space. It is important to understand that both EAGLE™ and EAXManager use the same coordinate system as DirectSound with positive Z forward, positive Y up, and positive X to the right. Any coordinates that do not conform to this system must first be converted before they are sent to EAXManager.

```
#include <dsound.h>
#include <eaxman.h>

BOOL SetSoundDynamicAttributes(LPEAXMANAGER eaxManager,
                               LPDIRECTSOUND3DBUFFER lpds3db,
                               float fXpos, float fYpos, float fZpos)
{
  // An EMPOINT struct is initialized with the sounds position. If the coordinate system
  // being used differed, here is where you would want to convert the data (e.g. swap Y and
  // Z if needed).
  EMPOINT ep, epv;
  ep.fX = fXpos; ep.fY = fYpos; ep.fZ = fZpos;

  long obstruction,occlusion;
  float obstructionLFRatio,occlusionLFRatio,occlusionRoomRatio;
  HRESULT hr = eaxManager->GetSourceDynamicAttributes(0,&ep,
                                                      &obstruction,&obstructionLFRatio,
                                                      &occlusion,&occlusionLFRatio,&occlusionRoomRatio,
                                                      &epv,0)
  if ( hr != EM_OK )
    return FALSE;

  // Get a propertyset from the buffer.
  LPKSPROPERTYSET lpksps;
  if ( lpds3db->QueryInterface(IID_IKsPropertySet, (LPVOID *)&lpksps) == DS_OK )
  {
    // First set a new virtual position for the sound.
    lpds3db->SetPosition(epv.fX, epv.fY, epv.fZ, DS3D_IMMEDIATE);

    lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OBSTRUCTION,
                NULL,0,&obstruction,sizeof(long));
    lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OBSTRUCTIONLFRATIO,
                NULL,0,&obstructionLFRatio,sizeof(float));
    lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSION,
                NULL,0,&occlusion,sizeof(long));
    lpksps->Set(DSPROPSETID_EAX_BufferProperties,DSPROPERTY_EAXBUFFER_OCCLUSIONLFRATIO,
                NULL,0,&occlusionLFratio,sizeof(float));
    lpksps->Set(DSPROPSETID_EAX_BufferProperties, DSPROPERTY_EAXBUFFER_OCCLUSIONROOMRATIO,
                NULL,0,&occlusionRoomRatio,sizeof(float));

    lpksps->Release();
    return TRUE;
  }

  return FALSE;
}
```

Appendix A: eaxManager COM interface

IEaxManager::GetDataSetSize

The **IEaxManager::GetDataSetSize** method returns the size in bytes of the currently loaded EAGLE™ environmental dataset

```
HRESULT GetDataSetSize(
      unsigned long *   ulsize  //Size in bytes of current data set
      DWORD             dwflags //Reserved for future use
);
```

Parameters

ulsize

[out] Size in bytes of currently loaded dataset.

dwflags

[in] Flags, reserved for future use.

Return Value

EM_OK.

Remarks

None.

IEaxManager::LoadDataSet

The **IEaxManager::LoadDataSet** method loads an EAGLE™ environmental dataset into EaxManager and sets it up for usage

```
HRESULT LoadDataSet(
      char *            szName  //ASCIIZ Filename of dataset to load
      DWORD             dwflags //0 or one of the available load flags
);
```

Parameters

szName

[in] ASCIIZ Filename of dataset to load.

dwflags

[in] Flags to use while getting attributes, currently the only flag defined is :

EMFLAG_LOADFROMMEMORY

Return Value

EM_OK.
EM_FILENOTFOUND.
EM_FILEINVALID.
EM_VERSIONINVALID.

Remarks

When using EMFLAG_LOADFROMMEMORY, the szName parameter must be a valid memory pointer addressing the memory image of the dataset file.

IEaxManager::FreeDataSet

The **IEaxManager::FreeDataSet** method unloads the currently loaded EAGLE™ environmental dataset

```
HRESULT FreeDataSet(
      DWORD             dwflags //Reserved for future use
);
```

Parameters

dwflags

[in] Flags, reserved for future use.

Return Value

EM_OK.

Remarks

None.

IEaxManager::GetListenerAttributes

The **IEaxManager::GetListenerAttributes** method gets the listener attributes from the loaded EAGLE™ environmental dataset

```
HRESULT GetListenerAttributes(
      LPLISTENERATTRIBUTES      lpAttributes    //Listener attributes
);
```

Parameters

lpAttributes

[out] Listener attribute structure, use this structure to set DirectSound3D Listener properties.

Return Value

EM_OK.

Remarks

None.

IEaxManager::GetSourceID

The **IEaxManager::GetSourceID** method returns the ID for a source given its name

```
HRESULT GetSourceID(
      char *            szName  //ASCIIZ name of source
      long *            ulID    //ID for source
);
```

Parameters

szName

[in] ASCIIZ name of source to get ID for.

ulID

[out] ID to use for source.

Return Value

EM_OK.
EM_IDNOTFOUND.

Remarks

None.

IEaxManager:: GetSourceAttributes

The **IEaxManager::GetSourceAttributes** method gets the attributes for a source given its ID

```
HRESULT GetSourceAttributes(
      long              ulID            //ID for source
      LPSOURCEATTRIBUTES lpAttributes    //Source attributes
);
```

Parameters

ulID

[in] ID to use for source.

lpAttributes

[out] Source attribute structure, use this structure to set DirectSound3D Buffer properties.

Return Value

EM_OK.
EM_INVALIDID.

Remarks

None.

IEaxManager:: GetSourceNumInstances

The **IEaxManager::GetSourceNumInstances** method gets the number of instances for a source ID

```
HRESULT GetSourceNumInstances(
      long              ulID            //ID for source
      long *            ulSourceInst    //Source instances
);
```

Parameters

ulID

[in] ID to use for source.

ulSourceInst

[out] Number of instances of the source placed in this geometry set.

Return Value

EM_OK.
EM_INVALIDID.

Remarks

None.

IEaxManager:: GetSourceInstancePos

The **IEaxManager::GetSourceInstancePos** method gets the 3D position of an instance for a source ID

```
HRESULT GetSourceInstancePos(
      long              ulID            //ID for source
      long              ulInstance      //Instance to query
      LPEMPOINT         lpPosition      //3D position of instance
);
```

Parameters

ulID

[in] ID to use for source.

ulInstance

[in] Instance to use for source.

LPEMPOINT lpPosition

[out] 3D position of instance (*ulInstance)* of source (*ulID)*.

Return Value

EM_OK.
EM_INVALIDID.

Remarks

None.

IEaxManager::GetEnvironmentID

The **IEaxManager::GetEnvironmentID** method returns the ID for an environment given its name

```
HRESULT GetEnvironmentID(
      char *            szName  //ASCIIZ name of environment
      long *            ulID    //ID for enviornemnt
);
```

Parameters

szName

[in] ASCIIZ name of environment to get ID for.

ulID

[out] ID to use for environment.

Return Value

EM_OK.
EM_IDNOTFOUND.

Remarks

None.

IEaxManager::GetEnvironmentAttributes

The **IEaxManager::GetEnvironmentAttributes** method gets the attributes for an environment given its ID

```
HRESULT GetEnvironmentAttributes(
      long                        ulID            //ID for environment
      LPEAXLISTENERPROPERTIES     lpAttributes    //Environment attributes
);
```

Parameters

ulID

[in] ID to use for environment.

lpAttributes

[out] Environment attribute structure, use this structure to set DirectSound3D EAXListener properties.

Return Value

EM_OK.
EM_INVALIDID.

Remarks

None.

IEaxManager::GetMaterialID

The **IEaxManager::GetMaterialID** method returns the ID for a material given its name

```
HRESULT GetMaterialID(
      char *            szName  //ASCIIZ name of material
      long *            ulID    //ID for material
);
```

Parameters

szName

[in] ASCIIZ name of material to get ID for.

ulID

[out] ID to use for material.

Return Value

EM_OK.
EM_IDNOTFOUND.

Remarks

None.

IEaxManager:: GetMaterialAttributes

The **IEaxManager::GetMaterialAttributes** method gets the attributes for a material given its ID

```
HRESULT GetMaterialAttributes(
      long                      ulID            //ID for material
      LPMATERIALATTRIBUTES      lpAttributes    //Material attributes
);
```

Parameters

ulID

[in] ID to use for material.

lpAttributes

[out] Material attribute structure, use this structure to set DirectSound3D EAXBuffer properties.

Return Value

EM_OK.
EM_INVALIDID.

Remarks

None.

IEaxManager::GetGeometrySetID

The **IEaxManager::GetGeometrySetID** method returns the ID for a geometry set given its name, because a single EAGLE™ dataset can contain multiple geometry sets

```
HRESULT GetGeometrySetID(
      char *            szName  //ASCIIZ name of geometry set
      long *            ulID    //ID for geometry
);
```

Parameters

szName

[in] ASCIIZ name of geometry set to get ID for.

ulID

[out] ID to use for geometry set.

Return Value

EM_OK.
EM_IDNOTFOUND.

Remarks

None.

IEaxManager::GetListenerDynamicAttributes

The **IEaxManager::GetListenerDynamicAttributes** method returns the attributes that change dynamically for the listener, given a geometry set and its position

```
HRESULT GetListenerDynamicAttributes(
      long               ulID    //ID for geometry
      LPEMPOINT          lpPos   //Position of listener
      unsigned long *    ulID    //ID for environment
      DWORD              dwFlags //Flags to use
);
```

Parameters

ulID

[in] ID of geometry set to use, if only one geometry set is used, specify zero.

lpPos

[in] Current position of 3D listener.

ulID

[out] ID to use for current environment.

dwFlags

[in] Flags to use while getting attributes, currently the only flag defined is :

EMFLAG_LOCKPOSITION

Return Value

EM_OK.
EM_IDNOTFOUND.

Remarks

Please note that both EAGLE and EaxManager use a left-handed coordinate system where positive Z point into the screen. This is the same coordinate system used by DirectSound3D, please make sure to apply proper conversion rules before calling this method.

The `IEaxManager::GetSourceDynamicAttributes` method relies on knowing the current listener position, initially this position will be the origin (0,0,0), for calculating values for occlusion, obstruction and finding the proper virtual position. The listenerposition used for the calculations is the one specfied when calling this method with the EMFLAG_LOCKPOSITION flag specfied.

This method only returns the ID for the environment in which the listener is currently located, the application has to take care of calling `IEaxManager::GetEnvironmentAttributes` to get the new listener attributes and passing them down to DirectSound3D.

IEaxManager::GetSourceDynamicAttributes

The **IEaxManager::GetSourceDynamicAttributes** method returns the attributes that change dynamically for a source, given a geometry set and its position

```
HRESULT GetSourceDynamicAttributes(
      long                      ulID            //ID for geometry
      LPEMPOINT                 lpPos           //Position of source
      long                      ulObstruction   //Obstruction
      float                     flObstructionLF //Obstruction LF
      long                      ulOcclusion     //Occlusion
      float                     flOcclusionLF   //Occlusion LF
      float                     flOcclusionRM   //Occlusion Room
      LPEMPOINT                 lpVirtualPos    //Source virtual pos
      DWORD                     dwFlags         //Flags to use
);
```

Parameters

ulID

[in] ID of geometry set to use, if only one geometry set is used, specify zero.

lpPos

[in] Current position of 3D sound source.

ulObstruction

[out] Source obstruction attribute, use this value to set DirectSound3D Buffer properties.

flObstructionLF

[out] Source obstruction lf ratio attribute, use this value to set DirectSound3D Buffer properties.

ulOcclusion

[out] Source occlusion attribute, use this value to set DirectSound3D Buffer properties.

flOcclusionLF

[out] Source occlusion lf ratio attribute, use this value to set DirectSound3D Buffer properties.

flOcclusionRM

[out] Source occlusion room ratio attribute, use this value to set DirectSound3D Buffer properties.

lpVirtualPos

[out] Virtual position to use for positioning sound, currently not implemented.

dwFlags

[in] Flags to use while getting attributes.

Return Value

EM_OK.
EM_IDNOTFOUND.

Remarks

Please note that both EAGLE and EaxManager use a left-handed coordinate system where positive Z point into the screen. This is the same coordinate system used by DirectSound3D, please make sure to apply proper conversion rules before calling this method.

This method relies on knowing the current listener position, initially this position will be the origin (0,0,0), for calculating values for occlusion, obstruction and finding the proper virtual position. The listener position used for the calculations is the one specfied when calling `IEaxManager::GetListenerDynamicAttributes` with the EMFLAG_LOCKPOSITION flag specfied.

This method only returns the new source attributes, the application has to take care of passing them down to DirectSound3D.