**就要K歌声卡控制方案**

Ver:1.0 2012-5-23

**1. Win7/Vista变更默认设备**

鉴于修改注册表的方法在Win7下需要管理员权限，这里介绍一种用代码来实现的方法。WIn7系统设计的本意对于这种操作是禁止的，因此没有公开的程序接口来变更，但是在目前的Win7/Vista下有一个未公开的COM接口能实现这个操作。需要注意的是，使用这个接口的风险就是不能保证将来操作系统更新之后该接口是否还有效。

接口名：

IPolicyConfig

具体代码：

// ----------------------------------------------------------------------------

// PolicyConfig.h

// Undocumented COM-interface IPolicyConfig.

// Use for set default audio render endpoint

// @author EreTIk

// ----------------------------------------------------------------------------

#pragma once

interface DECLSPEC\_UUID("f8679f50-850a-41cf-9c72-430f290290c8")

IPolicyConfig;

class DECLSPEC\_UUID("870af99c-171d-4f9e-af0d-e63df40c2bc9")

CPolicyConfigClient;

// ----------------------------------------------------------------------------

// class CPolicyConfigClient

// {870af99c-171d-4f9e-af0d-e63df40c2bc9}

//

// interface IPolicyConfig

// {f8679f50-850a-41cf-9c72-430f290290c8}

//

// Query interface:

// CComPtr[IPolicyConfig] PolicyConfig;

// PolicyConfig.CoCreateInstance(\_\_uuidof(CPolicyConfigClient));

//

// @compatible: Windows 7 and Later

// ----------------------------------------------------------------------------

interface IPolicyConfig : public IUnknown

{

public:

virtual HRESULT GetMixFormat(

PCWSTR,

WAVEFORMATEX \*\*

);

virtual HRESULT STDMETHODCALLTYPE GetDeviceFormat(

PCWSTR,

INT,

WAVEFORMATEX \*\*

);

virtual HRESULT STDMETHODCALLTYPE ResetDeviceFormat(

PCWSTR

);

virtual HRESULT STDMETHODCALLTYPE SetDeviceFormat(

PCWSTR,

WAVEFORMATEX \*,

WAVEFORMATEX \*

);

virtual HRESULT STDMETHODCALLTYPE GetProcessingPeriod(

PCWSTR,

INT,

PINT64,

PINT64

);

virtual HRESULT STDMETHODCALLTYPE SetProcessingPeriod(

PCWSTR,

PINT64

);

virtual HRESULT STDMETHODCALLTYPE GetShareMode(

PCWSTR,

struct DeviceShareMode \*

);

virtual HRESULT STDMETHODCALLTYPE SetShareMode(

PCWSTR,

struct DeviceShareMode \*

);

virtual HRESULT STDMETHODCALLTYPE GetPropertyValue(

PCWSTR,

const PROPERTYKEY,

PROPVARIANT \*

);

virtual HRESULT STDMETHODCALLTYPE SetPropertyValue(

PCWSTR,

const PROPERTYKEY,

PROPVARIANT \*

);

virtual HRESULT STDMETHODCALLTYPE SetDefaultEndpoint(

\_\_in PCWSTR wszDeviceId,

\_\_in ERole eRole

);

virtual HRESULT STDMETHODCALLTYPE SetEndpointVisibility(

PCWSTR,

INT

);

};

interface DECLSPEC\_UUID("568b9108-44bf-40b4-9006-86afe5b5a620")

IPolicyConfigVista;

class DECLSPEC\_UUID("294935CE-F637-4E7C-A41B-AB255460B862")

CPolicyConfigVistaClient;

// ----------------------------------------------------------------------------

// class CPolicyConfigVistaClient

// {294935CE-F637-4E7C-A41B-AB255460B862}

//

// interface IPolicyConfigVista

// {568b9108-44bf-40b4-9006-86afe5b5a620}

//

// Query interface:

// CComPtr[IPolicyConfigVista] PolicyConfig;

// PolicyConfig.CoCreateInstance(\_\_uuidof(CPolicyConfigVistaClient));

//

// @compatible: Windows Vista and Later

// ----------------------------------------------------------------------------

interface IPolicyConfigVista : public IUnknown

{

public:

virtual HRESULT GetMixFormat(

PCWSTR,

WAVEFORMATEX \*\*

); // not available on Windows 7, use method from IPolicyConfig

virtual HRESULT STDMETHODCALLTYPE GetDeviceFormat(

PCWSTR,

INT,

WAVEFORMATEX \*\*

);

virtual HRESULT STDMETHODCALLTYPE SetDeviceFormat(

PCWSTR,

WAVEFORMATEX \*,

WAVEFORMATEX \*

);

virtual HRESULT STDMETHODCALLTYPE GetProcessingPeriod(

PCWSTR,

INT,

PINT64,

PINT64

); // not available on Windows 7, use method from IPolicyConfig

virtual HRESULT STDMETHODCALLTYPE SetProcessingPeriod(

PCWSTR,

PINT64

); // not available on Windows 7, use method from IPolicyConfig

virtual HRESULT STDMETHODCALLTYPE GetShareMode(

PCWSTR,

struct DeviceShareMode \*

); // not available on Windows 7, use method from IPolicyConfig

virtual HRESULT STDMETHODCALLTYPE SetShareMode(

PCWSTR,

struct DeviceShareMode \*

); // not available on Windows 7, use method from IPolicyConfig

virtual HRESULT STDMETHODCALLTYPE GetPropertyValue(

PCWSTR,

const PROPERTYKEY,

PROPVARIANT \*

);

virtual HRESULT STDMETHODCALLTYPE SetPropertyValue(

PCWSTR,

const PROPERTYKEY,

PROPVARIANT \*

);

virtual HRESULT STDMETHODCALLTYPE SetDefaultEndpoint(

\_\_in PCWSTR wszDeviceId,

\_\_in ERole eRole

);

virtual HRESULT STDMETHODCALLTYPE SetEndpointVisibility(

PCWSTR,

INT

); // not available on Windows 7, use method from IPolicyConfig

};

如何使用：

void SetDefaultAudioPlaybackDevice( LPCWSTR devID )

{

IPolicyConfigVista \*pPolicyConfig;

ERole reserved = eConsole;

HRESULT hr = CoCreateInstance( \_\_uuidof( CPolicyConfigVistaClient ),

NULL, CLSCTX\_ALL, \_\_uuidof( IPolicyConfigVista ), ( LPVOID\* )&pPolicyConfig );

if( SUCCEEDED( hr ) )

{

hr = pPolicyConfig->SetDefaultEndpoint( devID, reserved );

pPolicyConfig->Release();

}

return hr;

}

注：devID 为IMMDevice：：GetId的返回值

**2. Win7/Vista声卡硬件ByPass设置**

在声卡驱动支持设置麦克风输出的情况下，有一种直接利用程序来实现此类设置的方法，基本思路就是利用**IDeviceTopology**来遍历整个声卡的拓扑结构，找到麦克风输出的设备，变更该设备的输出音量及是否静音来达到我们需要的效果。

1. 通过IMMDeviceEnumerator：：EnumAudioEndpoints得到所有的RenderEndPoints
2. 遍历所有的RenderEndPoints，得到他们的IDeviceTopology接口
3. 通过IDeviceTopology：：GetConnector及IConnect：：GetConnectedTo得到Render设备的入口节点IConnector
4. 通过IConnector：：QueryInterface得到节点的IPart接口
5. 以此为根节点，使用IPart::EnumPartsIncoming树形遍历其下的所有节点，找到麦克风输出的音量及静音设备。

**3. WinXP/Win2000声卡硬件ByPass设置**

1.首先获取声卡播放设备线路。

memset(&mxl, 0, sizeof(MIXERLINE));

mxl.cbStruct = sizeof(MIXERLINE);

mxl.dwComponentType = MIXERLINE\_COMPONENTTYPE\_DST\_SPEAKERS;

mr = ::mixerGetLineInfo((HMIXEROBJ)hMixer, &mxl, MIXER\_OBJECTF\_HMIXER | MIXER\_GETLINEINFOF\_COMPONENTTYPE);

1. 可以得出该设备的线路总数，其中每一条线路都是一根硬件输出线路。

maxconn = mxl.cConnections;

1. 遍历所有的硬件输出线路，找到所需要的线路。

submxl.dwDestination = mxl.dwDestination;

submxl.dwSource = x; //遍历硬件输出线

mr = ::mixerGetLineInfo((HMIXEROBJ)hMixer, &submxl, MIXER\_OBJECTF\_HMIXER | MIXER\_GETLINEINFOF\_SOURCE);

1. 在线路上进行设置，设置方法和设置其它线路方法一致。