

# Re: WMEncoder

---

*Source:*

<http://www.tech-archive.net/Archive/Media/microsoft.public.windowsmedia.sdk/2008-02/msg00027.html>

---

- *From:* [Joe] <[Joe@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:Joe@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 8 Feb 2008 09:45:02 -0800
- 

Thanks a lot .

I succeed doing that using something I find in the WEB, I paste this piece of code , maybe someone in the future will need them :

```
#region Interop decls
internal enum WmVersion
{
    Version4 = 0x00040000,
    Version7 = 0x00070000,
    Version8 = 0x00080000,
    Version9 = 0x00090000
}

[StructLayout(LayoutKind.Sequential)]
internal class WmMediaType
{
    public Guid majortype;
    public Guid subtype;
    [MarshalAs(UnmanagedType.Bool)]
    public bool bFixedSizeSamples;
    [MarshalAs(UnmanagedType.Bool)]
    public bool bTemporalCompression;
    public uint lSampleSize;
    public Guid formattype;
    public IntPtr pUnk;
    public uint cbFormat;
    public IntPtr pbFormat;
}

[StructLayout(LayoutKind.Explicit)]
internal class WaveFormatEx
{
    [FieldOffset(0)]
    public short wFormatTag; /* format type */
    [FieldOffset(2)]
    public short nChannels; /* number of
channels (i.e. mono, stereo...) */
    [FieldOffset(4)]
```

```

public int nSamplesPerSec; /* sample rate */
[FieldOffset(8)]
public int nAvgBytesPerSec; /* for buffer
estimation */
[FieldOffset(12)]
public short nBlockAlign; /* block size of
data */
[FieldOffset(14)]
public short wBitsPerSample; /* Number of
bits per sample of mono data */
[FieldOffset(16)]
public short cbSize; /* The count in
bytes of the size of extra information (after cbSize) */
}

[ComVisible(true), ComImport,
Guid("E1CD3524-03D7-11d2-9EED-006097D2D7CF"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
internal interface INSSBuffer
{
[PreserveSig]
int GetLength(
[Out] out uint length);

[PreserveSig]
int SetLength(
[In] uint length);

[PreserveSig]
int GetMaxLength(
[Out] out uint length);

[PreserveSig]
int GetBuffer(
[Out] out IntPtr buffer);

[PreserveSig]
int GetBufferAndLength(
[Out] out IntPtr buffer,
[Out] out uint length);
}

/// <summary>
/// IWMSStreamConfig
/// </summary>
[ComVisible(true), ComImport,
Guid("96406BDC-2B2B-11d3-B36B-00C04F6108FF"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
internal interface IWMSStreamConfig
{
void GetStreamType(

```

```
[Out] out Guid pguidStreamType);

void GetStreamNumber(
[Out] out int pwStreamNum);

void SetStreamNumber(
[In] int wStreamNum);

void GetStreamName(
[Out, MarshalAs(UnmanagedType.LPWSTR)] StringBuilder
pwszStreamName,
[In, Out] ref int pcchStreamName);

void SetStreamName(
[In, MarshalAs(UnmanagedType.LPWSTR)] string pwszStreamName);

void GetConnectionName(
[Out, MarshalAs(UnmanagedType.LPWSTR)] StringBuilder
pwszInputName,
[In, Out] ref int pcchInputName);

void SetConnectionName(
[In, MarshalAs(UnmanagedType.LPWSTR)] string pwszInputName);

void GetBitrate(
[Out] out int pdwBitrate);

void SetBitrate(
[In] int pdwBitrate);

void GetBufferWindow(
[Out] out int pmsBufferWindow);

void SetBufferWindow(
[In] int msBufferWindow);
}

[ComVisible(true), ComImport,
Guid("96406BCE-2B2B-11d3-B36B-00C04F6108FF"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IWMMediaProps
{
void GetType(
[Out] out Guid pguidType);

void GetMediaType(
[Out] IntPtr pType,
[In, Out] ref uint pcbType);

void SetMediaType(
[In] WmMediaType pType);
```

```
}

[ComVisible(true), ComImport,
Guid("96406BD5-2B2B-11d3-B36B-00C04F6108FF"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IWMMInputMediaProps : IWMMediaProps
{
#region IWMMediaProps
new void GetType(
[Out] out Guid pguidType);

new void GetMediaType(
[Out] IntPtr pType,
[In, Out] ref uint pcbType);

new void SetMediaType(
[In] WmMediaType pType);
#endregion

void GetConnectionName(
[Out, MarshalAs(UnmanagedType.LPWStr)] StringBuilder pwszName,
[In, Out] ref int pcchName);

void GetGroupName(
[Out, MarshalAs(UnmanagedType.LPWStr)] StringBuilder pwszName,
[In, Out] ref int pcchName);
};

[ComVisible(true), ComImport,
Guid("96406BDD-2B2B-11d3-B36B-00C04F6108FF"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IWMMStreamList
{
void GetStreams(
[Out] out int pwStreamNumArray,
[In, Out] ref int pcStreams);

void AddStream(
[In] int wStreamNum);

void RemoveStream(
[In] int wStreamNum);
}

[ComVisible(true), ComImport,
Guid("96406BDE-2B2B-11d3-B36B-00C04F6108FF"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IWMMutualExclusion : IWMMStreamList
{
#region IWMMStreamList
```

```
new void GetStreams(  
[Out] out int pwStreamNumArray,  
[In, Out] ref int pcStreams);  
  
new void AddStream(  
[In] int wStreamNum);  
  
new void RemoveStream(  
[In] int wStreamNum);  
#endregion  
  
void GetType(  
[Out] out Guid pguidType);  
  
void SetType(  
[In] Guid guidType);  
}  
  
[ComVisible(true), ComImport,  
Guid("96406BDB-2B2B-11d3-B36B-00C04F6108FF"),  
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]  
internal interface IWMPProfile  
{  
void GetVersion(  
[Out] out WmVersion pdwVersion);  
  
void GetName(  
[Out, MarshalAs(UnmanagedType.LPWStr)] out string pwszName,  
[In, Out] ref int pcchName);  
  
void SetName(  
[In, MarshalAs(UnmanagedType.LPWStr)] string pwszName);  
  
void GetDescription(  
[Out, MarshalAs(UnmanagedType.LPWStr)] out string  
pwszDescription,  
[In, Out] ref int pcchDescription);  
  
void SetDescription(  
[In, MarshalAs(UnmanagedType.LPWStr)] string pwszDescription);  
  
void GetStreamCount(  
[Out] out int pcStreams);  
  
void GetStream(  
[In] int dwStreamIndex,  
[Out, MarshalAs(UnmanagedType.Interface)] out IWMPStreamConfig  
ppConfig);  
  
void GetStreamByNumber(  

```

Re: WMEncoder

```
[In] int wStreamNum,  
[Out, MarshalAs(UnmanagedType.Interface)] out IWMSStreamConfig  
ppConfig);
```

```
void RemoveStream(  
[Out, MarshalAs(UnmanagedType.Interface)] out IWMSStreamConfig  
ppConfig);
```

```
void RemoveStreamByNumber(  
[In] int wStreamNum);
```

```
void AddStream(  
[In, MarshalAs(UnmanagedType.Interface)] IWMSStreamConfig  
pConfig);
```

```
void ReconfigStream(  
[In, MarshalAs(UnmanagedType.Interface)] IWMSStreamConfig  
pConfig);
```

```
void CreateNewStream(  
[In] ref Guid guidStreamType,  
[Out, MarshalAs(UnmanagedType.Interface)] out IWMSStreamConfig  
ppConfig);
```

```
void GetMutualExclusionCount(  
[Out] out int pcME);
```

```
void GetMutualExclusion(  
[In] int dwMEIndex,  
[Out, MarshalAs(UnmanagedType.Interface)] out  
IWMMutualExclusion ppME);
```

```
void RemoveMutualExclusion(  
[In, MarshalAs(UnmanagedType.Interface)] IWMMutualExclusion  
pME);
```

```
void AddMutualExclusion(  
[In, MarshalAs(UnmanagedType.Interface)] IWMMutualExclusion  
pME);
```

```
void CreateNewMutualExclusion(  
[Out, MarshalAs(UnmanagedType.Interface)] out  
IWMMutualExclusion ppME);  
}
```

```
[ComVisible(true), ComImport,  
Guid("96406BD4-2B2B-11d3-B36B-00C04F6108FF"),  
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]  
internal interface IWMWriter
```

```
{  
void SetProfileByID(  
[In] ref Guid guidProfile);  
  
void SetProfile(  
[In] IWMPProfile pProfile);  
  
void SetOutputFilename(  
[In, MarshalAs(UnmanagedType.LPWStr)] string pwszFilename);  
  
void GetInputCount(  
[Out] out int pcInputs);  
  
void GetInputProps(  
[In] int dwInputNum,  
[Out] out IWMPInputMediaProps ppInput);  
  
void SetInputProps(  
[In] int dwInputNum,  
[In] IWMPInputMediaProps pInput);  
  
void GetInputFormatCount(  
[In] int dwInputNumber,  
[Out] out int pcFormats);  
  
void GetInputFormat(  
[In] int dwInputNumber,  
[In] int dwFormatNumber,  
[Out] out IWMPInputMediaProps pProps);  
  
void BeginWriting();  
  
void EndWriting();  
  
void AllocateSample(  
[In] int dwSampleSize,  
[Out] out INSSBuffer ppSample);  
  
void WriteSample(  
[In] int dwInputNum,  
[In] long cnsSampleTime,  
[In] int dwFlags,  
[In, MarshalAs(UnmanagedType.Interface)] INSSBuffer pSample);  
  
void Flush();  
}  
  
[ComVisible(true), ComImport,  
Guid("d16679f2-6ca0-472d-8d31-2f5d55aee155"),  
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
```

```
internal interface IWMPProfileManager
{
void CreateEmptyProfile(
[In] WmVersion dwVersion,
[Out, MarshalAs(UnmanagedType.Interface)] out IWMPProfile
ppProfile);

void LoadProfileByID(
[In] ref Guid guidProfile,
[Out, MarshalAs(UnmanagedType.Interface)] out IWMPProfile
ppProfile);

void LoadProfileByData(
[In, MarshalAs(UnmanagedType.LPWStr)] string pwszProfile,
[Out, MarshalAs(UnmanagedType.Interface)] out IWMPProfile
ppProfile);

void SaveProfile(
[In, MarshalAs(UnmanagedType.Interface)] IWMPProfile
pIWMPProfile,
[Out, MarshalAs(UnmanagedType.LPWStr)] string pwszProfile,
[In, Out] ref uint pdwLength);

void GetSystemProfileCount(
[Out] out uint pcProfiles);

void LoadSystemProfile(
[In] uint dwProfileIndex,
[Out, MarshalAs(UnmanagedType.Interface)] out IWMPProfile
ppProfile);
}

[ComVisible(true), ComImport,
Guid("A970F41E-34DE-4a98-B3BA-E4B3CA7528F0"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
internal interface IWMCCodecInfo
{
void GetCodecInfoCount(
[In] ref Guid guidType,
[Out] out uint pcCodecs);

void GetCodecFormatCount(
[In] ref Guid guidType,
[In] uint dwCodecIndex,
[Out] out uint pcFormat);

void GetCodecFormat(
[In] ref Guid guidType,
[In] uint dwCodecIndex,
[In] uint dwFormatIndex,
[Out, MarshalAs(UnmanagedType.Interface)] out IWMPStreamConfig
```

```
ppIStreamConfig);
};

internal class WmMediaTypeId
{
private WmMediaTypeId()
{
}

public static readonly Guid Audio = new
Guid("73647561-0000-0010-8000-00AA00389B71");
public static readonly Guid Video = new
Guid("73646976-0000-0010-8000-00AA00389B71");
}

internal class WmFormatId
{
public static readonly Guid WaveFormatEx = new
Guid("05589f81-c356-11ce-bf01-00aa0055595a");
}

internal class NativeMethods
{
private NativeMethods()
{
}

[DllImport("WMVCore.dll", EntryPoint = "WMCreateWriter",
SetLastError = true, CharSet = CharSet.Unicode, ExactSpelling = true,
CallingConvention = CallingConvention.StdCall)]
public static extern int WMCreateWriter([In,
MarshalAs(UnmanagedType.IUnknown)] object unk, [Out,
MarshalAs(UnmanagedType.Interface)] out IWMWriter writer);

[DllImport("WMVCore.dll", EntryPoint = "WMCreateProfileManager",
SetLastError = true, CharSet = CharSet.Unicode, ExactSpelling = true,
CallingConvention = CallingConvention.StdCall)]
public static extern int WMCreateProfileManager([Out,
MarshalAs(UnmanagedType.Interface)] out IWMProfileManager
ppProfileManager);
}

#endregion

"Alessandro Angeli" wrote:
```

From: "[Joe]"

Re: WMEncoder

I can't find the namespace and reference of IWMWriter :-(  
Can you help me with this issue?  
what should I install and what reference to add to my  
project ?

Have you read my message? <<<requires COM InterOp and the  
DirectShow.NET wrapper can only help with a part of it>>>.  
There is no off-the-shelf .NET support for the WMF objects  
so you have to create your own InterOp wrapper for them  
before you can use them in .NET. There is a sample in the  
SDK that shows where to start, then you can use  
DirectShow.NET to help you with some data structures because  
they are the same, while the rest you have to do on your  
own.

--

// Alessandro Angeli  
// MVP :: DirectShow / MediaFoundation  
// mvpnews at riseoftheants dot com  
// <http://www.riseoftheants.com/mmx/faq.htm>