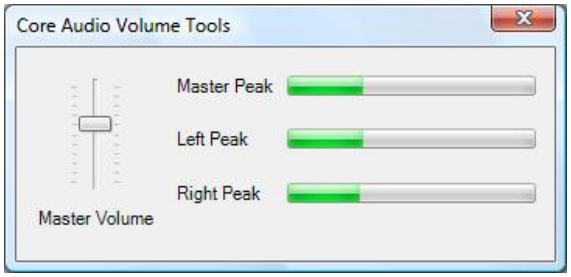
Vista Core Audio API Master Volume Control

By **Ray M.**, 3 May 2012

★★★★★ 4.82 (68 votes)

Download sample - 18.8 KB

Download source - 64.13 KB



Introduction

Windows Vista features a completely new set of user-mode audio components that provide per application volume control. All legacy APIs such as the **waveXxx** and **mixer*Xxx*** functions and DirectSound have been rebuilt on top of these new components so that without any code changes all '*old*' applications support this new audio API. This is a great thing ***except*** when your application is reading or querying the master volume settings of the operating system because all of a sudden, you are no longer controlling the master volume of the operating system but only of your own application.

Core Audio API

The new API is COM based, and split into four sub APIs which roughly do the following:

- MMDevice API - This API allows enumeration and instancing of the available audio devices in the system.
- WASAPI - This API allows playback and recording of audio streams.
- DeviceTopology API - This API allows access to hardware features such as bass, treble, and auto gain control.
- EndpointVolume API - This API allows access to the Volume and Peak meters.

The `MMDevice` and `EndpointVolume` APIs are needed to control the master volume and mute settings, while the APIs themselves are a huge improvement since the *legacy* functions lack a nice managed interface, making working with them somehow an unpleasant experience in C#. Writing COM interop code is not for the novice and is error-prone, hence creating a wrapper to do all the plumbing for us seems a good idea.

Using the Code

You always start by creating the enumerator class, allowing you to find the device that you want, or just settle for the default device.

Collapse | Copy Code

```
MMDeviceEnumerator devEnum = new MMDeviceEnumerator();
MMDevice defaultDevice =
  devEnum.GetDefaultAudioEndpoint(EDataFlow.eRender, ERole.eMultimedia);
```

After you have a reference to an `MMDevice` object, toggling Mute is as easy as:

Collapse | Copy Code

```
defaultDevice.AudioEndpointVolume.Mute =
            !defaultDevice.AudioEndpointVolume.Mute;
```

Or getting the volume of the left channel:

Collapse | Copy Code

```
Console.WriteLine("Left Volume : {0}",
      defaultDevice.AudioEndpointVolume.Channels[0].VolumeLevelScalar);
```

Since the wrapper implements the complete `EndpointVolume` API, we can also get more advanced things such as peak meters for all channels:

Collapse | Copy Code

```
Console.WriteLine("Master Peak : {0}",
      defaultDevice.AudioMeterInformation.MasterPeakValue);
Console.WriteLine("Left   Peak : {0}",
      defaultDevice.AudioMeterInformation.PeakValues[0]);
Console.WriteLine("Right  Peak : {0}",
      defaultDevice.AudioMeterInformation.PeakValues[1]);
```

If you just want to be informed when somebody changes the volume settings, you can subscribe to the `OnVolumeNotification` event like this:

Collapse | Copy Code

```
defaultDevice.AudioEndpointVolume.OnVolumeNotification += new
    AudioEndpointVolumeNotificationDelegate(
       AudioEndpointVolume_OnVolumeNotification);
.
.
void AudioEndpointVolume_OnVolumeNotification(AudioVolumeNotificationData data)
{
    Console.WriteLine("New Volume {0}", data.MasterVolume);
    Console.WriteLine("Muted      {0}", data.Muted);
}
```

Points of interest

The documentation of the volume notification states it must be non-blocking. The client should never wait on a synchronization object during an event callback. Keep that in mind when write event handlers.

Although Vista does per-application audio settings, Microsoft has not open up the APIs to enumerate Sessions. So it will not be possible to make a *sndvol.exe*-like application.

Right now, only the **`MMDevice`** and **`EndpointVolume`** APIs are implemented; I hope to add the **`WASAPI`** in a future update.

History

- 2007.04.23 - Initial article
- 2010.05.17 - Updated source and sample

License

About the Author

Muting all microphones on WinXP
PracticeSharp (or Practice#) - A Utility for Practicing your Musical Instrument with Playback
Volume Manipulation Classes
Simple Mixer Control Wrapper
MusicDrawer: A Simple Music player using WMP SDK
Programming Audio Effects in C#
Cam Alarm 2.0 - Alarm system run from a web camera
Windows Mobile Call Silencer
How to convert between (most) audio formats in .NET
Cross Platform Microphone Audio Processing Utility In Qt
A simple record and playback volume control class
A Platform for Unrestricted Low-level Hardware Programming and Experiments (NakedCPU)
Voice synthesis with Microsoft SAPI
Speech Recognition for the Web
CWinamp - more than just a Winamp2 API wrapper