Utilisation de PowerShell pour Gérer le Volume Audio

Introduction

Dans cet article, nous allons explorer comment utiliser PowerShell pour gérer le volume audio sur votre système. Nous allons utiliser des interfaces COM pour interagir avec les périphériques audio. Voici un exemple de script PowerShell qui permet de régler le volume et de gérer le mode muet.

Module

```
Add-Type -TypeDefinition @'
using System.Runtime.InteropServices;

[Guid("5CDF2C82-841E-4546-9722-0CF74078229A"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IAudioEndpointVolume {
  // f(), g(), ... are unused COM method slots. Define these if you care
  int f(); int g(); int h(); int i();
  int SetMasterVolumeLevelScalar(float fLevel, System.Guid
pguidEventContext);
  int j();
  int GetMasterVolumeLevelScalar(out float pfLevel);
  int k(); int l(); int m(); int n();
  int SetMute([MarshalAs(UnmanagedType.Bool)] bool bMute, System.Guid
pguidEventContext);
  int GetMute(out bool pbMute);
}
[Guid("D666063F-1587-4E43-81F1-B948E807363F"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IMMDevice {
  int Activate(ref System.Guid id, int clsCtx, int activationParams, out
IAudioEndpointVolume aev);
}
[Guid("A95664D2-9614-4F35-A746-DE8DB63617E6"),
InterfaceType(ComInterfaceType.InterfaceIsIUnknown)]
interface IMMDeviceEnumerator {
  int f(); // Unused
  int GetDefaultAudioEndpoint(int dataFlow, int role, out IMMDevice
endpoint);
}
[ComImport, Guid("BCDE0395-E52F-467C-8E3D-C4579291692E")] class
MMDeviceEnumeratorComObject { }
```

```
public class Audio {
  static IAudioEndpointVolume Vol() {
    var enumerator = new MMDeviceEnumeratorComObject() as
IMMDeviceEnumerator;
    IMMDevice dev = null;
Marshal.ThrowExceptionForHR(enumerator.GetDefaultAudioEndpoint(/*eRender*/
0, /*eMultimedia*/ 1, out dev));
    IAudioEndpointVolume epv = null;
    var epvid = typeof(IAudioEndpointVolume).GUID;
    Marshal.ThrowExceptionForHR(dev.Activate(ref epvid, /*CLSCTX_ALL*/ 23,
0, out epv));
    return epv;
  }
  public static float Volume {
    get {float v = -1;
Marshal.ThrowExceptionForHR(Vol().GetMasterVolumeLevelScalar(out v)); return
v;}
    set {Marshal.ThrowExceptionForHR(Vol().SetMasterVolumeLevelScalar(value,
System.Guid.Empty));}
  }
  public static bool Mute {
    get { bool mute; Marshal.ThrowExceptionForHR(Vol().GetMute(out mute));
return mute; }
    set { Marshal.ThrowExceptionForHR(Vol().SetMute(value,
System.Guid.Empty)); }
  }
}
'@
```

Exemple de Script

```
PS C:\> [Audio]::Volume         # Check current volume (now about 10%)
0,09999999
PS C:\> [Audio]::Mute           # See if speaker is muted
False
PS C:\> [Audio]::Mute = $true   # Mute speaker
PS C:\> [Audio]::Volume = 0.75  # Set volume to 75%
PS C:\> [Audio]::Volume         # Check that the changes are applied
0,75
PS C:\> [Audio]::Mute
True
PS C:\>
```

Conclusion

Ce script PowerShell utilise des interfaces COM pour interagir avec les périphériques audio de votre système. Vous pouvez l'utiliser pour régler le volume et gérer le mode muet de manière programmatique.