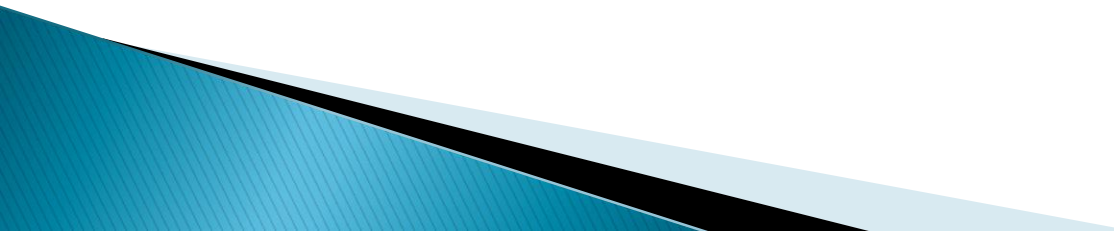


Die Windows Audio Session API

Eine Präsentation von Kevin Schaller

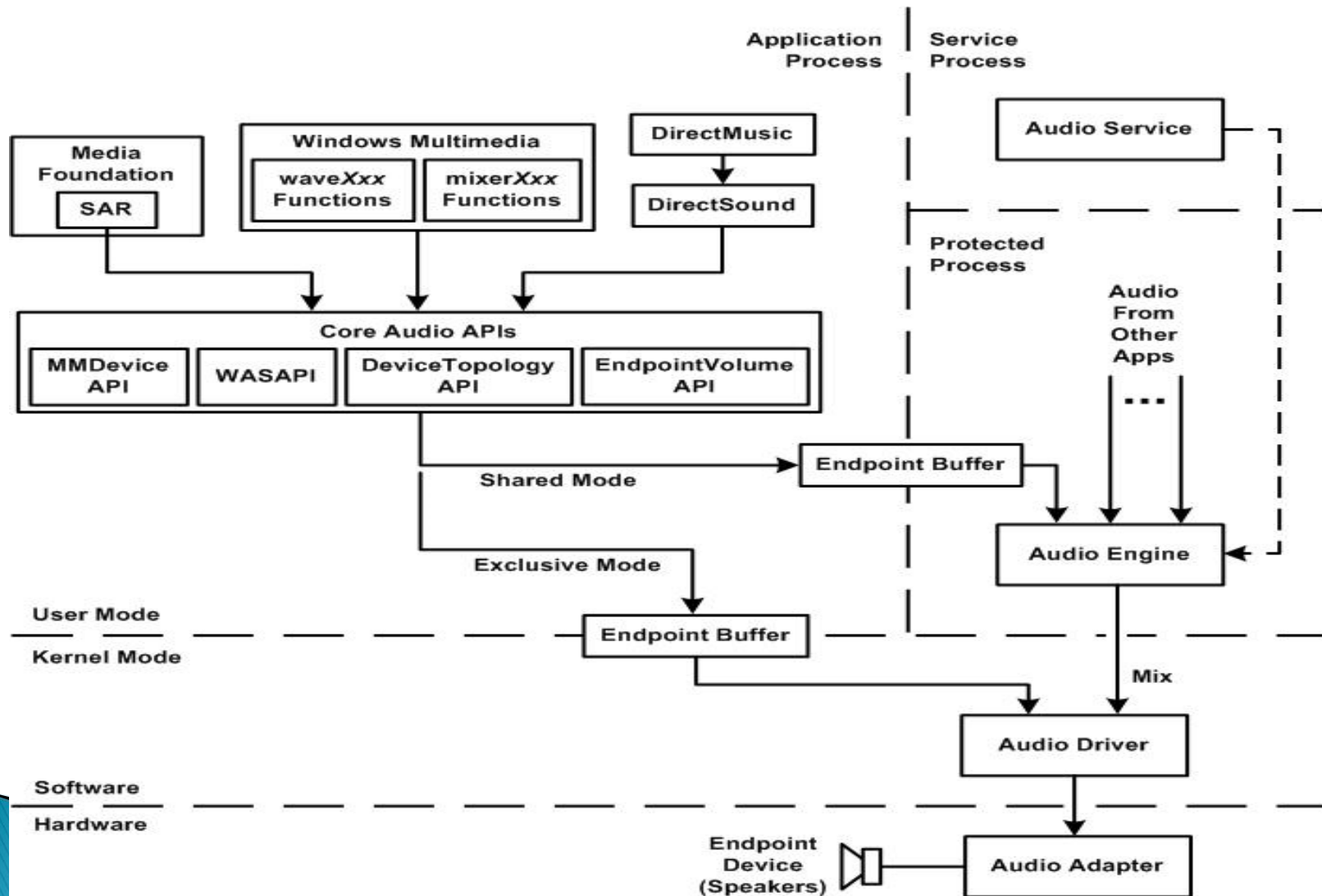
Inhaltsverzeichnis

- ▶ Überblick
 - ▶ Der Weg des Audio Streams
 - ▶ WASAPI – Pro/Contra
 - ▶ The missing Feature
 - ▶ Die API
 - ▶ Fazit
 - ▶ Sergej mit den GNU/Linux Audio Servern
- 

Überblick

- ▶ Win32 – Multimedia API
 - waveXXX
 - mixerXXX
- ▶ DirectX – DirectSound
- ▶ Media Foundation (SAR)
- ▶ Core Audio API's
 - Windows Audio Session API
 - Device Topology API
 - MMDevice API
 - EndpointVolume API

Der Weg des Audio Streams



WASAPI Pro/Contra

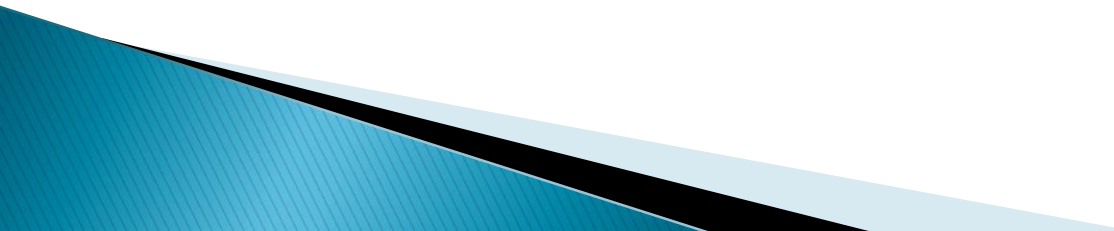
▶ Pro

- Umgeht die Windows Mixer
- Kann zwischen Shared und Exclusive Mode switchen
- Theoretische Latenz liegt bei 2ms
- Mit MMCSS unschlagbar schnell
- „Pro Audio“ Anwendungsorientiert (hardwarenah)

▶ Contra

- The missing Feature
- Erst ab Windows VISTA verfügbar

The missing Feature

- ▶ Sample Rate Converter
 - ▶ Komplizierte Implementierung
 - ▶ Großer Nachteil der WASAPI
 - ▶ Jedoch individuell anpassbar an Ansprüche
- 

Die API

```
REFERENCE_TIME hnsActualDuration;
IMMDeviceEnumerator *pEnumerator = NULL;
IMMDevice *pDevice = NULL;
IAudioClient *pAudioClient = NULL;
IAudioRenderClient *pRenderClient = NULL;
WAVEFORMATEX *pWfx = NULL;
UINT32 bufferFrameCount;
UINT32 numFramesAvailable;
UINT32 numFramesPadding;
BYTE *pData;
DWORD flags = 0;

CoCreateInstance(CLSID_MMDeviceEnumerator, NULL, CLSCTX_ALL,
                IID_IMMDeviceEnumerator, (void**) &pEnumerator);

pEnumerator->GetDefaultAudioEndpoint(eRender, eConsole, &pDevice);

pDevice->Activate(IID_IAudioClient, CLSCTX_ALL,
                 NULL, (void**) &pAudioClient);

pAudioClient->GetMixFormat(&pWfx);

pAudioClient->Initialize(AUDCLNT_SHAREMODE_SHARED, 0,
                       hnsRequestedDuration, 0, pWfx, NULL);

// Tell the audio source which format to use.
pMySource->SetFormat(pWfx);

// Get the actual size of the allocated buffer.
pAudioClient->GetBufferSize(&bufferFrameCount);

pAudioClient->GetService(IID_IAudioRenderClient,
                       (void**) &pRenderClient);
```

Die API

```
// Grab the entire buffer for the initial fill operation.
pRenderClient->GetBuffer(bufferFrameCount, &pData);

// Load the initial data into the shared buffer.
pMySource->LoadData(bufferFrameCount, pData, &flags);

pRenderClient->ReleaseBuffer(bufferFrameCount, flags);

// Calculate the actual duration of the allocated buffer.
hnsActualDuration = (double)REFTIMES_PER_SEC *
    bufferFrameCount / pWfx->nSamplesPerSec;

pAudioClient->Start(); // Start playing.

// Each loop fills about half of the shared buffer.
while (flags != AUDCLNT_BUFFERFLAGS_SILENT)
{
    // Sleep for half the buffer duration.
    Sleep((DWORD)(hnsActualDuration/REFTIMES_PER_MILLISEC/2));

    // See how much buffer space is available.
    pAudioClient->GetCurrentPadding(&numFramesPadding);

    numFramesAvailable = bufferFrameCount - numFramesPadding;

    // Grab all the available space in the shared buffer.
    pRenderClient->GetBuffer(numFramesAvailable, &pData);

    // Get next 1/2-second of data from the audio source.
    pMySource->LoadData(numFramesAvailable, pData, &flags);

    pRenderClient->ReleaseBuffer(numFramesAvailable, flags);
}
```

Fazit

- ▶ Sehr geeignet für Pro Audio Anwendungen
 - ▶ Unverzichtbar für VoIP-Communication
 - ▶ Sehr schwer zu implementieren für Laien
 - ▶ Übertrifft trotzdem die anderen Windows API's
- 