WDM音频组件

本节介绍系统提供的Microsoft Windows驱动程序模型（WDM）音频组件。这些是操作系统随附的软件组件。这些组件支持ISA，PCI和USB音频设备，并提供软件接口，应用程序可通过这些接口控制和监视这些设备。

译注:建议结合典型的WDM音频配置一文看

# 1. 用户模式WDM音频组件

用户模式Microsoft Windows驱动程序模型（WDM）音频组件是：

## WinMM系统组件

WinMM系统组件（Winmm.dll及其16位对应的Mmsystem.dll）实现Microsoft Windows多媒体API wave *Xxx*，midi *Xxx*，混合器*Xxx*和aux *Xxx*（请参阅Microsoft Windows SDK文档）。WinMM组件使用WDMAud系统驱动程序将WinMM API调用转换为内核流I / O请求。

## WDMAud系统驱动程序

用户模式WDMAud系统驱动程序（Wdmaud.drv）与内核模式WDMAud系统驱动程序（Wdmaud.sys）配对。WDMAud系统驱动程序一起在WinMM API调用和内核流I / O请求之间进行转换。内核模式WDMAud驱动程序是SysAudio系统驱动程序的客户端。

## DirectSound系统组件

DirectSound系统组件（Dsound.dll）支持DirectSound API（请参阅Microsoft Windows SDK文档）。DirectSound组件是SysAudio驱动程序的客户端。如果硬件混合可用，则SysAudio驱动程序将DirectSound硬件缓冲区直接连接到渲染设备。否则，SysAudio驱动程序会将DirectSound软件缓冲区连接到KMixer系统驱动程序。有关更多信息，请参见[使用DirectSound软件和硬件缓冲区渲染Wave内容](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/rendering-wave-content-using-directsound-software-and-hardware-buffers)。

## DirectMusic系统组件

DirectMusic系统组件（DMusic.dll）支持DirectMusic API（请参阅Microsoft Windows SDK文档）。该组件将对DirectMusic API的调用转换为对WDM音频设备的I / O请求。DirectMusic组件是SysAudio系统驱动程序的客户端。

## Windows音频服务

在Windows XP和更高版本中，Windows音频服务组件（Audiosrv.dll）管理基于Windows的程序的音频设备。停止Windows音频服务会阻止音频设备和效果正常运行。如果禁用了音频服务，则任何明显依赖它们的其他服务（包括WDM音频驱动程序）将无法启动。在Windows XP的家庭版，专业版和服务器版以及更高版本中，音频服务默认情况下配置为自动启动。但是，在Windows Server 2003及更高版本的Advanced Server，Data Center和Web Server版本中，默认情况下禁用音频服务。禁用音频服务后，安装音频设备不会启用它们-仅当管理员明确配置音频服务后，音频服务才能自动运行。**服务**对话框（在Windows控制面板中的“ **管理工具”**下查找）。

# 2. 内核模式WDM音频组件

内核模式Microsoft Windows驱动程序模型（WDM）音频组件是

## WDMAud系统驱动程序

内核模式WDMAud系统驱动程序（Wdmaud.sys）与用户模式WDMAud系统驱动程序（Wdmaud.drv）配对。这对WDMAud驱动程序在用户模式Microsoft Windows多媒体系统调用和内核流I / O请求之间进行转换。WDMAud为以下API执行I / O：**waveIn**，**waveOut**，**midiIn**，**midiOut**，**Mixer**和**aux**（在Microsoft Windows SDK文档中进行了描述）。内核模式WDMAud驱动程序是内核流（KS）filter，也是SysAudio系统驱动程序的客户端。

## SysAudio系统驱动程序

SysAudio系统驱动程序（Sysaudio.sys）构建用于渲染和捕获音频内容的filter图。SysAudio驱动程序将音频filter图表示为虚拟音频设备 (参考后续章节), 并将每个虚拟音频设备注册为KSCATEGORY\_AUDIO\_DEVICE设备接口的实例。（适配器驱动程序不应在SysAudio专用的此类别中进行注册。）例如，虚拟MIDI设备可能表示通过连接SWMidi驱动程序，KMixer驱动程序和端口/微型端口驱动程序创建的filter图。客户端仅与虚拟音频设备通信，而不与构成虚拟音频设备的单个设备通信。SysAudio驱动程序对客户端透明，它配置了filter图中的所有KS filter，这些filter连接在一起以形成虚拟音频设备。以下音频流源使用SysAudio构建的图形：

* DirectSound（请参阅Microsoft Windows SDK文档。）
* Windows多媒体API **waveIn**，**waveOut**，**midiIn**，**midiOut**，**Mixer**和**aux**（请参阅Windows SDK文档。）
* 红皮书(Redbook)CD数字音频（请参阅后续章节:红皮书系统驱动程序。）
* Sound Blaster仿真器（请参阅后续章节:SBEmul系统驱动程序。）
* 内核模式软件合成器（请参阅后续章节:SWMidi系统驱动程序和DMusic系统驱动程序。）
* DRMK系统驱动程序

## KMixer系统驱动程序

KMixer系统驱动程序（Kmixer.sys）是执行以下操作的KS filter：

* 混合多个PCM音频流
* 高质量格式转换
* 位深度转换
* 扬声器配置和声道映射

除了简单的8位和16位，单声道和立体声数据格式外，KMixer驱动程序还支持：

* PCM和IEEE浮点数据
* 位深度大于16位，并且具有两个以上通道的多通道格式
* 头部相关传递函数（HRTF）3-D处理

有关Windows各个版本中的音量范围和默认音量级别的信息，请参阅“ [默认音频音量设置”](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/default-audio-volume-settings)。

## 红皮书(Redbook)系统驱动程序

Redbook系统驱动程序（Redbook.sys）是KS filter，用于管理CD数字音频的渲染。Redbook驱动程序是SysAudio系统驱动程序的客户端。系统通过文件系统将CD数字音频路由到Redbook驱动程序，然后再路由到SysAudio驱动程序。CD数字音频在首选的WAVE输出设备（在“控制面板”的“多媒体”属性页中设置）上渲染。

## SBEmul系统驱动程序

SBEmul系统驱动程序（Sbemul.sys）为MS-DOS应用程序提供了Sound Blaster模拟。SBEmul驱动程序是SysAudio系统驱动程序的客户端。为了渲染和捕获内容，SysAudio驱动程序使用合适的wave和MIDI设备（在“控制面板”的“多媒体”属性页中进行设置）。

仅Windows 98 / Me支持Sound Blaster模拟。

## SWMidi系统驱动程序

SWMidi系统驱动程序（Swmidi.sys）是KS filter，可提供软件模拟的通用MIDI（General MIDI, GM）和高质量的Roland GS(Gerneral Standard/Sound)波表合成器。当硬件合成器不可用, **midiOut*X***xx应用程序使用SWMidi。SWMidi filter从WDMAud系统驱动器接收带有时间戳的MIDI流作为输入，并将PCM波流输出到KMixer系统驱动器。SWMidi在内部将所有声音混合在一起，以形成具有PCM WAVE格式的单个两通道输出流。

## DMusic系统驱动程序

DMusic系统驱动程序（Dmusic.sys）是KS filter，它支持软件仿真的高质量DLS（downloadable sound, 译注: 一种数字乐器声音文件格式/合成规范）合成器。DMusic驱动程序是系统提供的端口类微型端口驱动程序。它公开了一个DirectMusic引脚，该引脚支持[DirectMusic流数据范围](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/directmusic-stream-data-range)。DMusic filter从DirectMusic系统组件接收带有时间戳的MIDI流作为输入，并将PCM波流输出到KMixer系统驱动器。DMusic驱动程序在内部混合所有声音，以形成具有PCM WAVE格式的单个两通道输出流。DirectMusic应用程序必须显式选择内核模式软件合成器Dmusic.sys，以使用它代替DirectMusic的默认用户模式合成器。

## AEC系统驱动程序

AEC系统驱动程序（Aec.sys）通过在软件中实现AEC（回声消除）和噪声抑制算法来支持全双工DirectSound应用程序。有关更多信息，请参见[DirectSound Capture Effects](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/directsound-capture-effects)。

## DRMK系统驱动程序

DRMK系统驱动程序（Drmk.sys）是KS filter，用于解密包含受DRM保护的内容的音频流。有关更多信息，请参见[数字版权管理](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/digital-rights-management)。

## 分离器系统驱动程序

Splitter系统驱动程序（Splitter.sys）是KS filter，可从单个输入捕获流创建两个或更多输出流。分离器驱动程序透明地将输入流复制到另外两个输出流，而与输入流的格式无关。

Windows Me和Microsoft Windows XP及更高版本支持Splitter驱动程序。有关更多信息，请参见[AVStream Splitters](https://docs.microsoft.com/windows-hardware/drivers/stream/avstream-splitters)。

## 端口类适配器驱动程序和PortCls系统驱动程序

端口类适配器驱动(Port Class Audio Adapter)程序使用端口/微型端口驱动程序架构来支持音频设备。PortCls驱动程序包括对ISA和PCI音频设备的内置驱动程序支持。尽管PortCls系统驱动程序（Portcls.sys）还提供了供应商提供的端口类适配器驱动程序的框架，但是Microsoft建议供应商使用系统提供的端口类适配器驱动程序来支持ISA和PCI音频设备。PortCls框架还可用于为其他硬件总线上的音频设备或仅软件设备构造驱动程序。有关更多信息，请参见[端口类简介](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/introduction-to-port-class)。

## USB音频类系统驱动程序（Usbaudio.sys）

USB音频类系统驱动程序（Usbaudio.sys）是AVStream微型驱动程序，可为符合音频设备通用串行总线（USB）设备类定义的音频设备提供驱动程序支持。

[USB实施者论坛](https://go.microsoft.com/fwlink/p/?linkid=8780)网站上提供了“音频设备的USB设备类定义”规范（1.0版）。Usbaudio.sys支持USB音频规范中描述的功能的子集。。

在Windows 98中，Usbaudio.sys引入了对USB设备（如扬声器和麦克风）的支持。Windows Me中增加了对MIDI设备的支持。

在即插即用设备枚举过程中，如果音频设备将自身标识为兼容USB音频，则系统会自动加载USBAudio驱动程序以驱动该设备。USBAudio无需专用适配器驱动程序即可直接驱动设备。这意味着符合USB音频规范的设备不需要专有的适配器驱动程序。

Microsoft建议硬件供应商为其USB音频设备使用USBAudio驱动程序，而不是编写专有的适配器驱动程序。

在Windows 98中，USBAudio驱动程序支持以下功能：

* 所有I型格式（8位带符号PCM除外）
* AC-3 II型格式
* 同步类型: 同步和自适应
* 多通道设备

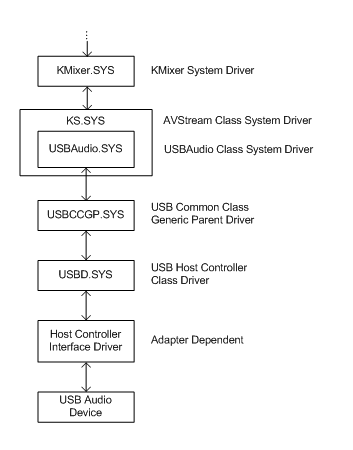
但是，Windows 98中的USBAudio不支持：

* 8位带符号PCM格式
* MPEG II型格式
* III型格式
* USB MIDI
* [**WAVEFORMATEXTENSIBLE**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ksmedia/ns-ksmedia-waveformatextensible) WAVE格式（USBAudio使用打包的WAVE\_FORMAT\_PCM代替24位数据。）

在Windows 98 Second Edition（SE），Windows Me和Windows 2000及更高版本中，USBAudio支持与Windows 98相同的所有功能，但有一个例外：USBAudio支持WAVEFORMATEXTENSIBLE，但不支持打包的24位数据WAVE\_FORMAT\_PCM。

在Windows Me和Windows XP及更高版本中，USBAudio支持Windows 98 SE和Windows 2000支持的所有功能。此外，Windows Me和Windows XP支持USB MIDI，但不支持USB MIDI Elements。

下图显示了USB音频设备的驱动程序层次结构。图中显示的所有驱动程序组件均由Microsoft随操作系统提供。



有关图中驱动程序组件的更多信息，请参见以下部分：

[AVStream概述](https://docs.microsoft.com/windows-hardware/drivers/stream/avstream-overview)

[系统提供的USB驱动程序](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/index)

## AVCAudio类系统驱动程序

AVCAudio类系统驱动程序（Avcaudio.sys）是AVStream微型驱动程序，可为位于IEEE 1394总线上的音频设备提供驱动程序支持。Windows XP和更高版本中提供了AVCAudio驱动程序以及对IEEE 1394音频设备的相关支持。

要使用系统提供的驱动程序，硬件供应商应设计其音频设备，使其符合以下规格的相应部分：

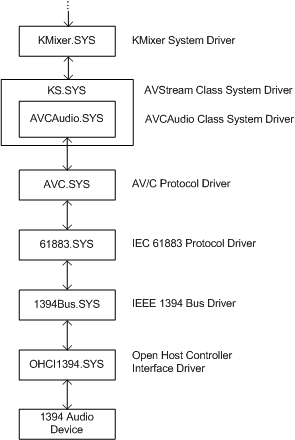
* IEC 61883-1和IEC 61883-6（IEC 60958）
* AV / C数字接口命令集通用规范版本 3.0
* AV / C音频子单元规范1.0
* 连接和兼容性管理规范1.0
* AV / C媒体流格式信息和协商
* 支持最新的AV / C音频子单元规格

这些规格可从[1394行业协会](https://go.microsoft.com/fwlink/p/?linkid=8728)网站上获得。AVCAudio驱动程序支持这些规范中描述的功能的子集。

当在即插即用设备枚举过程中音频设备将自身标识为符合IEEE 1394的音频设备时，系统会自动加载AVCAudio驱动程序以驱动该设备。AVCAudio无需专用适配器驱动程序即可直接驱动设备。这意味着符合适当的IEEE 1394规范的设备不需要专有的适配器驱动程序。

Microsoft建议硬件供应商为其IEEE 1394音频设备使用AVCAudio驱动程序，而不是编写专有的适配器驱动程序。

下图显示了Windows XP中IEEE 1394音频设备的驱动程序层次结构。在Windows XP和更高版本中，该图所示的所有驱动程序组件均由Microsoft随操作系统提供。



有关图中驱动程序组件的更多信息，请参见以下部分：

[AVStream概述](https://docs.microsoft.com/windows-hardware/drivers/stream/avstream-overview)

[AV / C客户端驱动程序](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/_stream/index)

[IEEE 1394总线](https://developer.microsoft.com/windows/hardware)

# 3. Wave和DirectSound组件

应用程序依赖于用户模式和内核模式组件的组合来捕获（输入）和渲染（输出）WAVE流。WAVE流是一种数字音频流，其数据格式由[**WAVEFORMATEX**](https://docs.microsoft.com/windows/desktop/api/mmreg/ns-mmreg-twaveformatex)或**[WAVEFORMATEXTENSIBLE](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ksmedia/ns-ksmedia-waveformatextensible)**结构描述。

应用程序可以使用以下两个软件接口之一进行WAVE渲染和捕获：

Microsoft Windows多媒体waveOut*Xxx*和waveIn*Xxx*函数

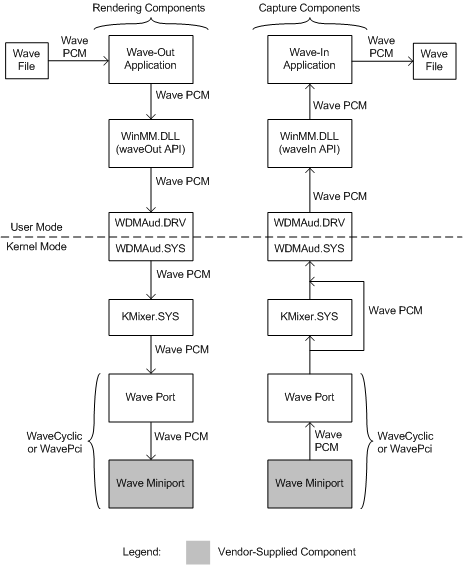
DirectSound和DirectSoundCapture API

waveOut*Xxx*和waveIn*Xxx*函数的行为基于旧版WAVE驱动器和设备的功能。从Windows 98开始，[WDMAud系统驱动程序将对](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "wdmaud_system_driver)这些功能的调用转换为对WDM音频驱动程序的命令。但是，通过模拟较旧的软件和硬件的行为，waveOut*Xxx*函数牺牲了3D声音效果和硬件加速，现在可以通过DirectSound API获得这些功能。有关DirectSound和Windows Multimedia wave函数的更多信息，请参见Microsoft Windows SDK文档。

DirectSound和Windows Multimedia wave函数是[SysAudio系统驱动程序的](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "sysaudio_system_driver)客户端，该[驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components#sysaudio_system_driver)将构建用于处理wave和DirectSound流的音频filter图。图形构建对于使用这些软件接口的应用程序是透明的。

## Wave组件

下图显示了wave应用程序用来渲染或捕获由wave PCM数据组成的数字音频流的用户模式和内核模式组件。



渲染组件显示在上图的左侧，捕获组件显示在右侧。表示wave微型端口驱动程序的框被涂黑，以表示它们是供应商提供的组件。图中的其他组件是系统提供的。

在该图的左上方，wave渲染（或“ wave-out”）应用程序通过waveOut*Xxx*函数与WDM音频驱动程序接口，该函数在用户模式[WinMM系统组件](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "winmm_system_component) Winmm.dll中实现。该应用程序从文件中读取WAVE音频样本的块，然后调用**[waveOutWrite](https://docs.microsoft.com/previous-versions/dd743876(v=vs.85))**函数来渲染它们。

由用户模式和内核模式组件（Wdmaud.drv和Wdmaud.sys）组成的WDMAud缓冲来自**[waveOutWrite](https://docs.microsoft.com/previous-versions/dd743876(v=vs.85))**调用的WAVE数据，并将WAVE流输出到[KMixer系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "kmixer_system_driver)，该[驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components#kmixer_system_driver)显示在图中的WDMAud下方。

KMixer是一个系统组件，它从一个或多个源接收wave PCM流，并将它们混合在一起以形成单个输出流，该输出流也采用wave PCM格式。

KMixer将波流输出到WaveCyclic或WavePci设备，其端口和微型端口驱动程序显示在上图左侧的KMixer下方。微型端口驱动程序将自身绑定到端口驱动程序，以形成代表基础音频渲染设备的wave filter。典型的渲染设备输出模拟信号，该信号驱动一组扬声器或一个外部音频单元。渲染设备还可以通过S / PDIF连接器输出数字音频。有关WaveCyclic和WavePci的更多信息，请参阅[Wave Filters](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/wave-filters)。

另外，KMixer可以将其输出流传递到USB音频设备，该设备由[USBAudio类系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "usbaudio_class_system_driver)（未显示）控制，而不是WaveCyclic或WavePci设备。

适配器驱动程序通过**[分别](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)**使用**GUID**值**CLSID\_PortWaveCyclic**或**CLSID\_PortWavePci**调用**[PcNewPort](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)**来创建WaveCyclic或WavePci端口驱动程序的实例。

上图的右侧显示了支持将WAVE数据捕获到文件的应用程序所需的组件。Wave-capture（或“ wave-in”）应用程序通过waveIn *Xxx*函数与WDM音频驱动程序进行通信，该函数在WinMM系统组件中实现。

在该图的右下角，WAVE捕获装置由微型端口和端口驱动程序控制。端口驱动器和微型端口驱动程序（可以是WaveCyclic或WavePci类型）绑定在一起以形成代表捕获设备的filter。该设备通常从麦克风或其他音频源捕获模拟信号，并将其转换为Wave PCM流。设备可能还会通过S / PDIF连接器输入数字音频流。

WAVE端口驱动程序将其WAVE流直接输出到KMixer或WDMAud。如果需要在WDMAud接收之前进行采样率转换，则该流必须通过KMixer。如图所示，执行音频流的同时渲染和捕获的系统可能需要KMixer的两个实例。请注意，SysAudio会根据需要自动创建这些实例。

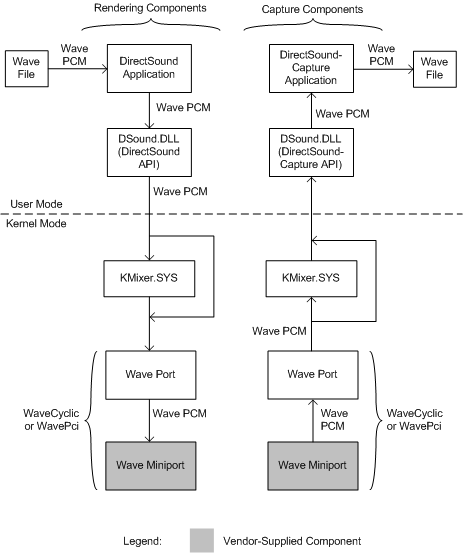
或者，捕获的WAVE源可以是USB音频设备，而不是WaveCyclic或WavePci设备。在这种情况下，USBAudio驱动程序（未显示）将流传递到KMixer。

不管波流是被USB设备捕获，还是被WaveCyclic或WavePci设备捕获，KMixer都会在需要时对流执行采样率转换，但不会与其他流混合。KMixer将结果流输出到Wdmaud.sys（WDMAud系统驱动程序的内核模式一半）。用户模式的一半Wdmaud.drv通过waveIn*Xxx*函数将WAVE流输出到应用程序，该函数在Winmm.dll中实现。最后，在该图的顶部，wave-capture应用程序将wave数据写入文件。

在wave-capture应用程序调用**[waveInOpen](https://docs.microsoft.com/previous-versions/dd743847(v=vs.85))**函数打开捕获流时，它将传入指向其回调例程的指针。当发生WAVE捕获事件时，操作系统将使用包含捕获设备的下一个WAVE样本块的缓冲区调用回调例程。作为对回调的响应，应用程序将下一个wave数据块写入文件。

## DirectSound组件

下图显示了DirectSound应用程序用来渲染或捕获wave数据的用户模式和内核模式组件。



渲染组件显示在上图的左半部分，捕获组件显示在右侧。Wave微型端口驱动程序显示为黑色方框，表明它们是供应商提供的组件。图中的其他组件是系统提供的。

在图的左上方，DirectSound应用程序将WAVE数据从文件加载到用户模式[DirectSound系统组件](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "directsound_system_component)（Dsound.dll）管理的声音缓冲区。该组件将波流发送到WaveCyclic或WavePci设备，其端口和微型端口驱动程序显示在该图的左下方。如果设备上有硬件混音器引脚，则流将绕过KMixer直接传递到Wave端口驱动器。否则，流首先通过KMixer，后者将其与其他同时播放的流混合。KMixer将混合流输出到端口驱动程序。

和以前一样，微型端口驱动程序将自身绑定到端口驱动程序，以形成代表基础音频渲染设备的wave filter。例如，此设备可能通过一组扬声器播放流。

或者，可以通过USB音频设备而不是WaveCyclic或WavePci设备来渲染波流。在这种情况下，流无法绕过KMixer。USBAudio类系统驱动程序（未显示）始终将流传递给KMixer。

上图的右侧显示了支持DirectSoundCapture应用程序的组件。该应用程序记录从WaveCyclic或WavePci捕获设备接收到的wave数据。该设备将来自麦克风的模拟信号转换为例如波流。该设备的wave端口和微型端口驱动程序出现在该图的右下角。如图所示，端口驱动程序从微型端口驱动程序接收流作为输入，并将其直接输出到用户模式DirectSound组件Dsound.dll，或通过KMixer间接输出。这取决于捕获设备是否提供了硬件捕获引脚。

或者，捕获WAVE的源可以是USB音频设备。在这种情况下，流无法绕过KMixer。USBAudio驱动程序（未显示）始终将流传递到KMixer。

如果将KMixer插入捕获流的路径，则会在需要时对流执行采样率转换，但不会与其他流混合。

在上图的右上角，应用程序从DirectSoundCapture缓冲区读取wave数据并将其写入文件。

# 4. MIDI和DirectMusic组件

应用程序依赖于用户和内核模式组件的组合来捕获和播放MIDI和DirectMusic流。

应用程序可以使用以下两个软件接口之一来进行MIDI回放和捕获：

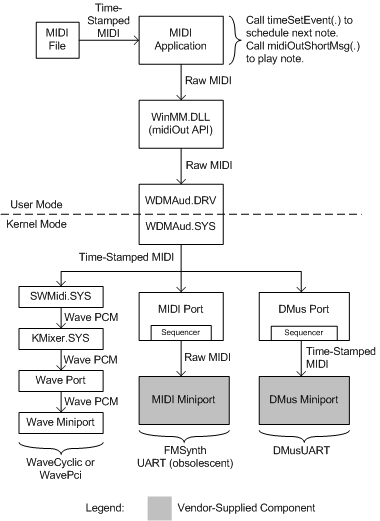
* Microsoft Windows多媒体**midiOut***Xxx*和**midiIn***Xxx*功能
* DirectMusic API

**midiOut***Xxx*和**midiIn***Xxx*函数的行为基于旧版MIDI驱动程序和设备的功能。从Windows 98开始，[WDMAud系统驱动程序将对](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "wdmaud_system_driver)这些功能的调用转换为对WDM音频驱动程序的命令。但是，通过模拟较旧的软件和硬件的行为，**midiOut***Xxx*和**midiIn***Xxx*函数牺牲了精确定时和增强功能，这些功能现在可通过DirectMusic API获得。有关DirectMusic和Windows Multimedia MIDI功能的更多信息，请参见Microsoft Windows SDK文档。

DirectMusic和Windows Multimedia MIDI功能是[SysAudio系统驱动程序的](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "sysaudio_system_driver)客户端，该[驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components#sysaudio_system_driver)将构建用于处理MIDI和DirectMusic流的音频filter图。对于使用这些软件接口的应用程序, 图形构建是透明的。

## MIDI组件

下图显示了MIDI应用程序用来*播放* MIDI数据的用户模式和内核模式组件。该应用程序通过**midiOut***Xxx*函数与WDM音频驱动程序接口，该函数在[WinMM系统组件](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "winmm_system_component) Winmm.dll中实现。



上图中的MIDI应用程序从MIDI文件中读取带时间戳的MIDI事件并进行播放。MIDI和DMus微型端口驱动程序显示为黑色方框，表示它们可以是供应商提供的组件。如果合适，供应商可能会选择使用系统提供的微型端口驱动程序之一-FMSynth，UART或DMusUART，而不是编写自定义微型端口驱动程序。图中的所有其他组件都是系统提供的。

典型的MIDI播放应用程序的主循环调用**timeSetEvent**安排下一个音符开启或音符关闭事件。该调用将指向应用程序回调例程的函数指针作为其参数之一。当事件发生并且操作系统调用回调例程时，此例程将调用**midiOutShortMsg**来打开或关闭一个或多个调度任务。该**midiOutShortMsg**功能存储在页面锁定的数据缓冲区的MIDI信息，以避免在通话过程中载入页面(译注:为了保证实时性)。有关**timeSetEvent**和**midiOutShortMsg**调用的更多信息，请参见Microsoft Windows SDK文档。

WDMAud由用户和内核模式组件（Wdmaud.drv和Wdmaud.sys）组成，它记录来自**midiOutShortMsg**调用的原始MIDI消息到达的时间。WDMAud将这些时间戳与MIDI消息结合在一起，以生成MIDI流，并将其发送到图中WDMAud下方的内核模式组件之一。

为MIDI应用程序构建音频filter图时，SysAudio仅选择上图中出现的三个可能的连接(到SWMidi，MIDI端口或DMus端口驱动程序)之一。如果应用程序选择默认的MIDI设备，则SysAudio首先会寻找其MIDI或DMus微型端口驱动程序具有MIDI引脚的合成器设备。如果在注册表中找不到此类设备，则SysAudio会改用[SWMidi系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "swmidi_system_driver)（Swmidi.sys）。SWMidi是一种KS filter，可在软件中实现波表合成器，并且仅需要可以渲染WAVE音频流的设备。

SWMidi将所有声音混合在一起以产生单WAVE PCM流，然后将其输出到[KMixer系统驱动器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "kmixer_system_driver)。KMixer依次将PCM格式的波流传递到WaveCyclic或WavePci设备，其端口和微型端口驱动程序出现在该图的左下角。另外，KMixer可以将其输出流传[递到](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "usbaudio_class_system_driver)由[USBAudio类系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components#usbaudio_class_system_driver)（未显示）控制的USB音频设备。

在上图中，MIDI端口驱动程序从WDMAud获取带时间戳的MIDI流，并将其转换为原始MIDI消息，MIDI微型端口驱动程序通过合成器设备播放这些消息。MIDI端口驱动程序包含一个音序器，该音序器通过软件实现，并能够以1毫秒的计时器分辨率安排原始MIDI消息。

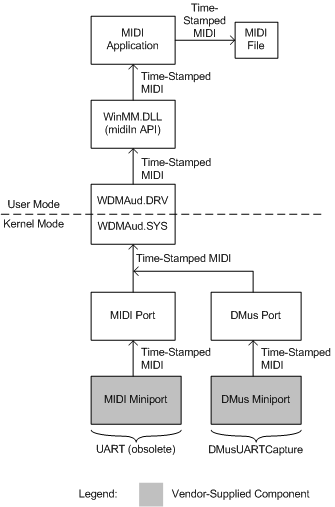
如果合成器设备包含硬件定序器(sequencer)，则DMus端口驱动程序可以获得比MIDI端口驱动程序高得多的定时精度。在这种情况下，DMus微型端口驱动程序应指定一个足够大的硬件缓冲区，以吸收由于与ISR（中断服务程序）和其他高优先级操作争夺CPU时间而导致的抖动。DMus端口驱动程序输出到微型端口驱动程序的MIDI流中的时间戳是具有100纳秒分辨率的64位值。(译注: Dmusic端口驱动程序可以支持硬件定序器, 而MIDI端口驱动程序不支持).

如果DMusic合成器没有硬件定序器，则它必须依赖DMus端口驱动程序的软件定序器，与MIDI端口驱动程序的软件定序器一样，计时器分辨率为一毫秒。

适配器驱动程序通过分别使用GUID值**CLSID\_PortMidi**或**CLSID\_PortDMus**调用**[PcNewPort来](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)**创建MIDI或DMus端口驱动程序。在Windows XP和更高版本中，MIDI和DMus端口驱动程序共享相同的软件实现。

在上图的底部显示的是系统提供的微型端口驱动程序FMSynth，UART和DMusUART的名称，这些名称包含在Portcls.sys中。适配器驱动程序通过调用**[PcNewMiniport](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewminiport)**创建这些微型端口驱动程序之一。FMSynth和UART提供[IMiniportMidi](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportmidi)接口，而DMusUART提供[IMiniportDMus](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/nn-dmusicks-iminiportdmus)接口。请注意，UART现在已过时（在Windows 98 Gold之后），并且仅受现有驱动程序支持。相反，新的适配器驱动程序应使用DMusUART（在Windows 98 SE和更高版本以及Windows 2000和更高版本中），该实现了UART功能的超集。DMusUART是不支持DLS下载或硬件定序器的DMus微型端口驱动程序的示例。Windows驱动程序工具包（WDK）中的示例音频驱动程序中提供了FMSynth和DMusUART微型端口驱动程序的源代码。

下图显示了MIDI应用程序用来*捕获* MIDI数据的用户模式和内核模式组件。该应用程序通过**midiIn***Xxx*功能连接到WDM音频驱动程序。



在上图中，MIDI和DMus微型端口驱动程序显示为黑色方框，表示它们可以是供应商提供的组件。如果合适，供应商可能会选择使用系统提供的微型端口驱动程序之一UART或DMusUARTCapture。图中的所有其他组件都是系统提供的。

MIDI数据的来源通常是MPU-401设备。通过调用**PcNewMiniport**，适配器驱动程序可以创建系统提供的微型端口驱动程序之一UART或DMusUARTCapture，以从MPU-401设备捕获MIDI数据。同样，UART已过时，新的驱动程序应该使用DMusUARTCapture（在Windows 98 SE和更高版本以及Windows 2000和更高版本中）。

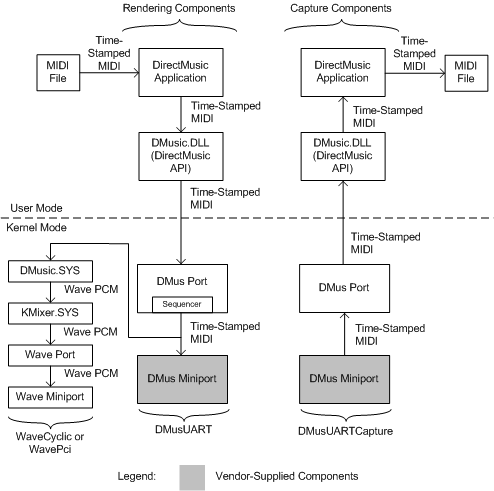
每次发生MIDI音符打开或关闭事件时，MIDI或DMusic捕获微型端口驱动程序（在上图的底部）会在MIDI消息上添加时间戳，然后再将其添加到流到MIDI的MIDI流中或DMus端口驱动程序。

MIDI或DMusic捕获端口驱动程序将带时间戳的MIDI流输出到Wdmaud.sys（WDMAud系统驱动程序的内核模式一半）。用户模式下的一半Wdmaud.drv通过**midiIn***Xxx*函数将带时间戳的MIDI流输出到应用程序，该函数在Winmm.dll中实现。

图顶部的MIDI应用程序将带有时间戳的MIDI事件写入MIDI文件。在应用程序调用**midiInOpen**打开MIDI输入流时，它会将函数指针传递给其回调例程。发生音符打开或音符关闭事件时，操作系统将使用包含一个或多个带时间戳的MIDI消息的数据块调用回调例程。这些消息上的时间戳基本上与MIDI或DMus微型端口驱动程序最初生成的时间戳相同。

## DirectMusic组件

下图显示了DirectMusic应用程序用来*回放*或*捕获* MIDI数据的用户和内核模式组件。



播放组件显示在上图的左半部分，捕获组件显示在右侧。DMus微型端口驱动程序显示为黑色方框，表示它们可以是供应商提供的组件。如果合适，供应商可以改用系统提供的微型端口驱动程序之一DMusUART或DMusUARTCapture。图中的其他组件是系统提供的。

在该图的左上角，DirectMusic应用程序将带有时间戳的MIDI流从文件定向到用户模式[DirectMusic系统组件](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "directmusic_system_component)（DMusic.dll），后者又将流定向到DMus端口驱动程序。如果有此驱动程序，可以将其绑定到DirectMusic合成器或MPU-401设备的微型端口驱动程序。或者，可以将端口驱动程序绑定到[DMusic系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "dmusic_system_driver)（Dmusic.sys），后者是系统提供的DMus微型端口驱动程序，可以在软件中实现支持DLS的波表合成器，并且只需要可以渲染WAVE的设备即可。音频流。

像SWMidi一样，DMusic驱动程序Dmusic.sys将其所有声音混合在一起以生成单个PCM格式的波流，然后将其输出到KMixer。反过来，KMixer可以将波流传递到WAVE设备，该设备的端口和微型端口驱动程序显示在该图的左下角，或传递到由USBAudio系统驱动程序控制的USB音频设备，该设备不会出现在图中。

DirectMusic捕获组件出现在上图的右半部分。该图右下角的DMusic捕获微型端口驱动程序控制捕获硬件并为其记录的每个MIDI消息加时间戳。DMus端口驱动程序将带时间戳的MIDI流定向到用户模式DirectMusic组件DMusic.dll。应用程序通过DirectMusic API访问此流，并将带有时间戳的MIDI数据写入文件。

适配器驱动程序可以使用系统提供的DMusUARTCapture微型端口驱动程序来控制MPU-401捕获设备。适配器驱动程序通过使用GUID值**CLSID\_DMusUARTCapture**调用**PcNewMiniport来**创建此微型端口驱动程序。生成的微型端口驱动程序对象支持**IMiniportDMus**接口。Windows驱动程序工具包（WDK）中的示例音频驱动程序中提供了DMusUARTCapture微型端口驱动程序的源代码。

如果**愿意，** DirectMusic应用程序也可以通过诸如SWMidi（Swmidi.sys）之类的**midiOut***Xxx*设备运行。为简单起见，上图中省略了此路径。DMusic驱动程序（Dmusic.sys）需要初始DLS下载才能正确运行。使用SWMidi可以避免此要求。

# 5. 虚拟音频设备

虚拟音频设备代表渲染和捕获音频内容的filter图。系统音频驱动程序（SysAudio）使用可用的硬件和软件组件来确定要构建的filter图。

有关系统音频驱动程序的更多信息，请参见[SysAudio系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components#sysaudio_system_driver)。

SysAudio的客户端包括DirectSound和[WDMAud系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "wdmaud_system_driver)，它们充当WDM音频驱动程序与特定音频的Microsoft Windows多媒体API waveIn，waveOut，midiIn，midiOut，混合器和aux之间的接口（在Microsoft Windows SDK文档中进行了介绍）。

Windows驱动程序工具包（WDK）中的[KsStudioStudio实用](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/ksstudio-utility)工具是绕过SysAudio并允许用户手动构造filter图的应用程序示例。

在PnP设备枚举之后，SysAudio会评估已注册的音频硬件和软件组件，以确定如何构造其客户端可能需要的各种音频filter图。

在确定可以从可用的硬件和软件组件构建的filter图表列表之后，SysAudio将这些图表注册为虚拟音频设备，以进行播放，录制，MIDI输入/输出和混合。SysAudio为其虚拟音频设备专门保留注册表类别KSCATEGORY\_AUDIO\_DEVICE。适配器驱动程序不应在此类别中注册自己。

SysAudio客户端可以像对待硬件或软件组件的filter工厂一样对待虚拟音频设备的filter工厂。当客户端要求实例化虚拟设备上的特定引脚时，SysAudio会自动构造图形，并透明地管理图形的内部引脚对客户端的连接。这允许客户端将filter图视为单个filter，从而避免了图管理（如filter通信）的复杂性。