**音频过滤器，引脚和节点**

* 2017/04/20
* 2分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/audio-filters--pins--and-nodes.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/audio-filters--pins--and-nodes.md" \o "1个贡献者)

Microsoft Windows驱动程序模型（WDM）适配器驱动程序将其音频硬件公开为筛选器工厂的集合，每个筛选器工厂都可以创建一个或多个筛选器实例。内核流（KS）过滤器对象可以封装音频硬件功能，该功能对流过过滤器的波形音频数据执行某种类型的数字处理。例如，过滤器可能会进行流的渲染或合成，或者可能在流中添加混响。

过滤器实例公开了引脚工厂，每个引脚工厂都可以创建一个或多个引脚实例。这些引脚可以连接到其他滤波器的引脚，以生成滤波器图。要成为音频过滤器图表的一部分，过滤器必须具有一个或多个引脚实例。

引脚表示输入或输出连接点，数据流通过输入或输出连接点进入或退出过滤器。每个引脚指定了它可以支持的数据格式范围，并且只有具有兼容格式的流可以流过该引脚。

WDM音频设备的过滤器以节点和连接的形式公开其内部拓扑。

拓扑节点位于通过过滤器的数据路径上。节点表示过滤器内的控制点。每个节点在逻辑上封装过滤器功能的模块块，并对通过该节点的数据流执行数字信号处理。节点可能代表例如可以在软件控制下进行调节的音量控制。

过滤器对象还指定其各个引脚和节点之间的连接。这些连接中隐含的是沿着通过过滤器的每个数据路径的节点顺序。

本节介绍了特定于WDM音频驱动器的滤波器，引脚和节点的功能。讨论了以下主题：

[音频过滤器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filters)

[过滤工厂](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/filter-factories)

[引脚工厂](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/pin-factories)

[节点和连接](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/nodes-and-connections)

[音频过滤器图](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filter-graphs)

[波形滤波器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/wave-filters)

[MIDI和DirectMusic滤波器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/midi-and-directmusic-filters)

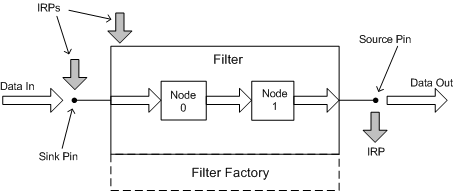
[拓扑过滤器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/topology-filters)

有关内核流过滤器，引脚和节点的更一般性讨论，请参见[KS Minidriver Architecture](https://docs.microsoft.com/windows-hardware/drivers/stream/ks-minidriver-architecture)。

**音频过滤器**

* 2017/04/20
* 2分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/audio-filters.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/audio-filters.md" \o "1个贡献者)

KS过滤器是内核对象，由内核对象句柄标识。在下图中，中间的大框是代表音频设备的KS滤波器。数据流从左侧流入过滤器，经过几个节点进行处理，然后从右侧离开过滤器。过滤器由过滤器工厂创建，该工厂显示为一个框，在过滤器的底部带有虚线边缘。



在图中，滤波器上实例化了两个引脚。左侧的引脚是数据接收器，右侧的引脚是数据源。数据通过漏极引脚流入滤波器，并通过源极引脚流出滤波器。

按照惯例，KS中术语源和宿的使用以缓冲区为中心（或者更一般地，以连接为中心）。一个滤波器上的源极引脚连接到另一个滤波器的漏极引脚时，通常需要一个数据缓冲器。缓冲器消除了数据从源引脚到达并进入宿引脚的速率的不规则性。（当然，并不是所有的连接都需要缓冲。例如，在同一适配器卡上的两个设备之间可能会发生无缓冲的连接，例如，接收和接收数据速率更容易匹配。）

相反，用于SRC和DST（源和目标）混合器行的混合器API术语是以设备为中心的：

* 流通过SRC混合器管线进入混合器设备。
* 流通过DST混合器管线离开混合器设备。

换句话说，SRC混频器线映射到KS滤波器上的数据接收器引脚，而DST混频器线映射到数据源引脚。有关更多信息，请参见[内核流拓扑到音频混合器API的转换](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-streaming-topology-to-audio-mixer-api-translation)。

为了简单起见，该图省略了过滤器的引脚工厂，后者创建了引脚实例。

引脚和过滤器除了是数据接收器和数据源以外，还可以是IRP接收器和IRP源。引脚和过滤器不仅可以接收IRP，而且还可以发送IRP。图中的三个黑色箭头表示IRP。图左侧的引脚是IRP接收器。右侧的引脚是IRP源。该图还显示了IRP发送到过滤器对象本身。

**过滤工厂**

* 2017/04/20
* 3分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/filter-factories.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/filter-factories.md" \o "1个贡献者)

音频适配器驱动程序提供了过滤器工厂来管理过滤器的实例化。每个过滤器工厂都可以实例化一个或多个特定类型的KS过滤器。如果过滤器类型封装了特定的硬件功能，则工厂可以实例化的该类型的过滤器数量受基础硬件资源的限制。

因为过滤器工厂管理着很大程度上独立的硬件功能块，所以每个过滤器工厂本身都可以视为设备驱动程序。实际上，在上一段中使用的术语适配器驱动程序是指一组相关的驱动程序（过滤器工厂），这些驱动程序打包在一起以管理适配器卡上的各种硬件功能。

与任何其他Microsoft Windows驱动程序模型（WDM）驱动程序一样，过滤器工厂处理电源管理和设置功能。在安装过程中，驱动程序的INF文件注册一个或多个过滤器设备名称（请参阅[设备标识字符串](https://docs.microsoft.com/windows-hardware/drivers/install/device-identification-strings)）。此过程将名称加载到系统注册表中，并将每个过滤器工厂与一个或多个KS过滤器类别相关联，如[安装音频适配器的设备接口中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/installing-device-interfaces-for-an-audio-adapter)所述。所有音频设备都归类为KSCATEGORY\_AUDIO，但是音频设备也可能归类为其他类别，例如KSCATEGORY\_RENDER（对于音频渲染设备）或KSCATEGORY\_CAPTURE（对于音频捕获设备）。驱动程序通过其为该设备注册过滤器的各种类别来宣传设备的一般功能。例如，当[SysAudio系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "sysaudio_system_driver)需要特定类型的音频设备时，它将在注册表中查找属于适当类别的设备。

操作系统使用[设备安装组件中](https://docs.microsoft.com/windows-hardware/drivers/install/system-provided-device-installation-components)所述的Setup API，以发现并枚举注册表中的所有KSCATEGORY\_AUDIO筛选器工厂。每个工厂的注册表项都指定了过滤器工厂的友好名称及其设备名称，这是客户端传递给实例化过滤器的创建文件调用的长字符串。可以从内核模式对**[ZwCreateFile](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ntifs/nf-ntifs-ntcreatefile)**或**CreateFile**进行此调用从用户模式。过滤器是内核模式对象，由内核句柄标识。create-file调用返回一个实例句柄，客户端可以使用该实例句柄来引用过滤器。音频图中的用户模式客户端或上游过滤器可以使用此句柄将IOCTL请求发送或转发到过滤器。有关**CreateFile的**更多信息，请参见Microsoft Windows SDK文档。

例如，典型的WDM音频适配器卡可能位于PCI总线上，并包含多个I / O连接器，用于呈现或捕获波形数据。此卡上的单个音频设备可能包含用于驱动一组扬声器和输出电缆的模拟音频输出插孔，以及用于从麦克风和输入电缆接收信号的模拟音频输入插孔。WDM音频系统将设备表示为过滤器，并将音频插孔表示为该过滤器上的引脚。

音频设备的筛选器实现为单独的端口和微型端口驱动程序，这些驱动程序绑定在一起以共同起作用：

* 微型端口驱动程序包含特定于硬件的代码。
* 端口驱动程序包含特定类型的所有过滤器通用的通用代码。

供应商编写了微型端口驱动程序，其中包含过滤器管理音频硬件所需的所有专有代码。操作系统提供了端口驱动程序，可以通过PortCls系统驱动程序（Portcls.sys；请参阅[端口类适配器驱动程序和PortCls系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "port_class_adapter_driver_and_portcls_system_driver)）进行访问。将过滤器实现分为端口驱动程序和微型端口驱动程序，简化了为专有设备编写驱动程序的任务。

过滤器工厂实例化过滤器时，它首先为过滤器创建微型端口驱动程序对象。然后，过滤器工厂将创建适当端口对象的实例，并将微型端口驱动程序对象绑定到该实例，以形成功能全面的过滤器。[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。端口和微型端口驱动程序通过定义明确的软件接口相互通信。有关这些接口的更多信息，请参见[微型端口接口](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/miniport-interfaces)和[支持设备](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/supporting-a-device)。

音频过滤器将底层音频设备的结构公开为引脚工厂，节点和内部连接的集合。微型端口驱动程序将这些信息合并到过滤器描述符中，该描述符是[**PCFILTER\_DESCRIPTOR**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/ns-portcls-pcfilter_descriptor)类型的结构。反过来，此结构包含滤波器的引脚工厂，节点和内部连接的各个描述符。这些描述符是以下类型的结构：

**[PCPIN\_DESCRIPTOR](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/ns-portcls-pcpin_descriptor)**

[**PCNODE\_DESCRIPTOR**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/ns-portcls-pcnode_descriptor)

[**PCCONNECTION\_DESCRIPTOR**](https://docs.microsoft.com/previous-versions/windows/hardware/drivers/ff537688(v=vs.85))

要从微型端口驱动程序获取过滤器描述符，端口驱动程序将调用**[IMiniport :: GetDescription](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iminiport-getdescription)**方法。

有关驱动程序如何设置其PCFILTER\_DESCRIPTOR结构的示例，请参见Windows驱动程序工具包（WDK）中sb16示例音频驱动程序中的头文件Table.h。

**引脚工厂**

* 2017/04/20
* 3分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/pin-factories.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/pin-factories.md" \o "1个贡献者)

音频滤波器的引脚工厂描述了滤波器可以实例化的所有引脚。如前所述，音频微型端口驱动程序将引脚信息存储在**[PCPIN\_DESCRIPTOR](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/ns-portcls-pcpin_descriptor)**结构的数组中。每个结构都指定一个引脚工厂，并通过其在数组中的索引来标识引脚工厂。该索引通常称为*引脚ID*。

PCPIN\_DESCRIPTOR结构包含一个自动化表和一个**[KSPIN\_DESCRIPTOR](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ks/ns-ks-kspin_descriptor)**结构。

KSPIN\_DESCRIPTOR结构包含有关引脚工厂中引脚的以下信息：

* 过滤器相对数据流的方向
* 筛选器相对于通讯流的方向（在所有当前的Windows版本中，KS筛选器均使用IRP进行通讯。）
* 引脚类别
* 友好名称
* 实例功能
* 数据格式功能

结构的“ **类别”**和“ **名称”**成员指定引脚工厂的引脚类别和友好名称。对于过滤器中的每个引脚工厂，微型端口驱动程序指定**类别**和**名称** GUID 的组合，以共同唯一地标识引脚工厂。如果两个或多个引脚工厂共享相同的“ **类别”**值，则每个引脚工厂都有一个“ **名称”**值，以区别于其他。如果只有单个引脚工厂具有特定的**Category**值，则该值足以标识该引脚工厂，并且该引脚工厂的**Name**值可以设置为**NULL**。有关编码示例，请参见[公开过滤器拓扑](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/exposing-filter-topology)。有关引脚类别的信息，请参见[引脚类别属性](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/pin-category-property)。

引脚工厂指定它作为扩展**[KSDATARANGE](https://docs.microsoft.com/previous-versions/ff561658(v=vs.85))**结构数组支持的数据格式范围：

* 支持其输入或输出流一系列wave或DirectSound数据格式的引脚工厂指定了**[KSDATARANGE\_AUDIO](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ksmedia/ns-ksmedia-ksdatarange_audio)**结构数组。
* 支持输入或输出流的MIDI或DirectMusic数据格式范围的引脚工厂指定了**[KSDATARANGE\_MUSIC](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ksmedia/ns-ksmedia-ksdatarange_music)**结构数组。

KSDATARANGE\_AUDIO和KSDATARANGE\_MUSIC是KSDATARANGE的扩展版本。有关两种类型的数据范围的示例，请参见[音频数据格式和数据范围](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-data-formats-and-data-ranges)。

在将一个过滤器上的漏极引脚连接到另一个过滤器上的源引脚之前，图形构建器（例如[SysAudio系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "sysaudio_system_driver)）可以在数据范围内搜索兼容格式。图构建器通常调用过滤器的[数据交叉处理程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/data-intersection-handlers)，该[处理程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/data-intersection-handlers)允许过滤器本身选择兼容的格式。

一个过滤器可以有多个引脚工厂，一个引脚工厂可以支持多个引脚实例。

* 在过滤器上具有多个引脚工厂对于区分流经过滤器的不同类型数据的单独数据路径很有用。例如，一个引脚工厂可能支持PCM数据流，而另一个引脚工厂可能支持AC-3流。
* 单个过滤器可以支持同时渲染和捕获流。渲染和捕获路径具有不同的过滤器工厂集。
* 在宿引脚工厂上具有多个引脚实例通常意味着混合，在这种情况下，过滤器包含一个SUM节点（**[KSNODETYPE\_SUM](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-sum)**）。

像过滤器一样，引脚是内核对象，并由内核句柄标识。引脚实例的句柄是通过调用**[KsCreatePin](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ks/nf-ks-kscreatepin)**创建的。作为内核对象，可以将引脚指定为IRP的目标。当向引脚发送IOCTL请求时，驱动程序的客户端指定引脚句柄。

建立[音频过滤器图表时](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filter-graphs)，SysAudio通过连接它们的引脚将一个过滤器链接到另一个过滤器。一个滤波器的源极引脚可以连接到另一滤波器的漏极引脚。来自源引脚的数据和IRP通过此连接流到宿引脚。为了建立连接，图构建器（通常是SysAudio）首先通过调用**[KsCreatePin](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ks/nf-ks-kscreatepin)**创建源引脚，然后通过再次调用**KsCreatePin**创建宿引脚。但是，在第二个调用中，客户端指定新的接收器引脚将连接到在第一个调用中创建的源引脚。

**节点和连接**

* 2017/04/20
* 2分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/nodes-and-connections.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/nodes-and-connections.md" \o "1个贡献者)

该过滤器以节点描述符数组（**[PCNODE\_DESCRIPTOR](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/ns-portcls-pcnode_descriptor)**结构）的形式提供其拓扑节点的描述。数组中的每个描述符描述一个节点，并包含一个指定节点类型的GUID（例如**[KSNODETYPE\_REVERB](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-reverb)**）。有关为音频设备定义的标准节点类型的列表，请参见“ [音频拓扑节点”](https://docs.microsoft.com/windows-hardware/drivers/audio/audio-topology-nodes)。

过滤器通过描述符数组中节点的索引来标识其每个节点。例如，当向过滤器或过滤器上的特定引脚发送特定于节点的属性请求时，客户端会在请求中包含节点ID（数组索引），以便标识目标节点。

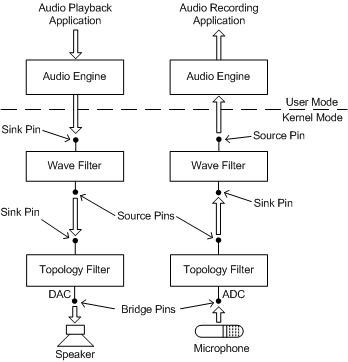
该过滤器以连接描述符数组（**[PCCONNECTION\_DESCRIPTOR](https://docs.microsoft.com/previous-versions/windows/hardware/drivers/ff537688(v=vs.85))**结构）的形式提供对其内部连接的描述。每个描述符描述过滤器的内部连接之一。描述符可以描述引脚和节点之间的连接，也可以描述两个节点之间的连接。

过滤器公开的节点和连接一起定义了过滤器的内部拓扑。拓扑是音频设备内部布局的映射，应准确反映其代表的硬件的组织。例如，Microsoft Windows Multimedia混音器API将过滤器的内部连接转换为混音器线，并将其节点转换为混音器线的控件（请参阅[内核流拓扑到音频混音器API的转换](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-streaming-topology-to-audio-mixer-api-translation)）。过滤器内部拓扑中的任何不正确之处都会反映在混合器行表示中，并且可能在使用混合器API的应用程序中导致错误或意外行为。

**音频过滤器图**

* 2017/04/20
* 2分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/audio-filter-graphs.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/audio-filter-graphs.md" \o "1个贡献者)

KS过滤器图是已连接在一起以处理一个或多个数据流的KS过滤器的集合。音频过滤器图是KS过滤器图，它由处理音频数据流的过滤器组成。例如，下图是执行音频渲染和捕获的音频滤波器图的简化图。



在图中，滤波器图从两个波形滤波器顶部的引脚延伸到两个拓扑滤波器底部的引脚。用户模式软件模块和外部音频设备（即扬声器和麦克风）位于图形外部。

图下半部分的四个滤波器表示音频适配器上的硬件设备，可以渲染和捕获波流。图中所示的每个过滤器都是通过将端口驱动程序绑定到微型端口驱动程序来实现的。适配器驱动程序通过将WaveRT，WavePci或WaveCyclic端口驱动程序绑定到相应的Wave *Xxx*微型端口驱动程序来形成波形滤波器。适配器驱动程序通过将拓扑端口驱动程序绑定到拓扑微型端口驱动程序来形成拓扑过滤器。

在图的左侧，来自DirectSound或waveOut应用程序的音频流（顶部）通过扬声器（底部）播放。在右侧，DirectSoundCapture或waveIn应用程序（顶部）记录了从麦克风输入的流（底部）。在两侧，音频引擎的实例插在波形滤波器和应用程序之间，该实例对Windows Vista中的系统执行混合。（在Windows Server 2003，Windows XP，Windows 2000和Windows Me / 98中，[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)是系统混合器。）

音频引擎是一种通用的软件过滤器，可以在用户模式下运行，并且可以在其源极和宿引脚上轻松地在各种音频格式和采样率之间进行转换。音频引擎通常可以容纳为硬件配置的流格式与应用程序期望的流格式之间的差异。

在上图的底部，驱动扬声器的源极引脚和接收麦克风信号的漏极引脚标记为桥接引脚。桥销桥接了过滤器图和外部世界之间的边界。

在上图中，每个波形滤波器及其对应的拓扑滤波器之间显示的数据路径通常表示一个物理连接：音频适配器上的固定硬件连接，无法通过软件配置。

因为桥式引脚或具有物理连接的引脚是永久连接的，所以该引脚隐式存在并且无法实例化或删除。因此，尽管您可以查询过滤器对象的桥接引脚的[KSPROPSETID\_Pin](https://docs.microsoft.com/windows-hardware/drivers/stream/kspropsetid-pin)属性，但是没有桥接引脚对象（桥接引脚的实例）可将IRP发送到该对象。相同规则适用于具有物理连接的引脚。

通过桥接引脚或物理连接的信号可以是模拟的也可以是数字的。

例如，在上图中，两个桥接引脚都处理模拟信号。左侧的桥接引脚传输来自DAC（数模转换器）的输出信号，该DAC驱动扬声器。右侧的桥脚接收来自麦克风的信号，该麦克风进入ADC（模数转换器）。但是，桥接引脚也可能代表音频设备上的S / PDIF连接器。在这种情况下，通过桥接引脚的信号是数字信号，而不是模拟信号。

**波形滤波器**

* 2018/05/08
* 8分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/wave-filters.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/wave-filters.md" \o "1个贡献者)

波形滤波器表示呈现和/或捕获波形格式的数字音频数据的设备。应用程序通常通过DirectSound API或Microsoft Windows多媒体waveOut *Xxx*和waveIn *Xxx*函数来访问这些设备的功能。有关WDM音频驱动程序可以支持的波形格式的信息，请参见**[WAVEFORMATEX](https://docs.microsoft.com/windows/desktop/api/mmreg/ns-mmreg-twaveformatex)**和**[WAVEFORMATEXTENSIBLE](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/ksmedia/ns-ksmedia-waveformatextensible)**。

甲*波渲染*滤波器接收作为输入的波数字音频流，并输出一个模拟音频信号（一组扬声器或外部混合器的）或数字音频流（到S / PDIF连接器，例如）。

甲*波捕获*过滤器接收作为输入的一个模拟音频信号（从麦克风或输入插孔）或数字流（从S / PDIF连接器，例如）。同一滤波器输出包含数字音频数据的波流。

单个滤波器可以同时执行渲染和捕获。例如，这种类型的过滤器可能代表一种音频设备，该设备可以通过一组扬声器播放音频，并同时通过麦克风记录音频。或者，如[动态音频子设备中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/dynamic-audio-subdevices)所述，可以将波形渲染和波形捕获硬件表示为单独的波形滤波器。

音频适配器驱动程序通过将wave微型端口驱动程序（该硬件供应商作为适配器驱动程序的一部分实现）与系统实现的wave端口驱动程序绑定在一起来形成wave滤波器。微型端口驱动程序处理波形滤波器的所有特定于硬件的操作，而端口驱动程序则管理所有通用的波形滤波器功能。

PortCls系统驱动程序（Portcls.sys）实现了三个Wave端口驱动程序：WaveRT，WavePci和WaveCyclic。

三种类型的滤波器的操作如下：

* 甲*WaveRT*滤波器分配用于波数据的缓冲器，使该缓冲器到用户模式客户直接访问。缓冲区可以由连续或不连续的内存块组成，具体取决于波形设备的硬件功能。客户端访问缓冲区作为虚拟内存的连续块。缓冲区是循环的，这意味着当设备的读取（用于渲染）或写入（用于捕获）指针到达缓冲区的末尾时，它将自动回绕到缓冲区的开头。
* 一个*WavePci*过滤器直接访问客户端的缓存。尽管客户端将缓冲区作为单个连续的虚拟内存块访问，但是WavePci筛选器必须将缓冲区作为一系列可能不连续的内存块访问。包含渲染或捕获流连续部分的块在设备处排队。当设备的读或写指针到达一个块的末尾时，它将移至队列中下一个块的开始。
* 甲*WaveCyclic*滤波器分配由存储器中，用于作为其输出（用于渲染）或输入（捕捉）缓冲单个的，连续的块的缓冲器。该缓冲区是循环的。由于客户端不能直接访问该缓冲区，因此驱动程序必须在驱动程序的循环缓冲区和客户端的用户模式缓冲区之间复制数据。

WaveRT优于WavePci和WaveCyclic。WavePci和WaveCyclic与早期版本的Windows一起使用。

WaveRT过滤器可以表示驻留在系统总线上的音频设备，例如PCI或PCI Express。WaveRT过滤器相对于WaveCyclic或WavePci过滤器的主要优势在于，WaveRT过滤器允许用户模式客户端直接与音频硬件交换音频数据。相反，WaveCyclic和WavePci滤波器都需要驱动程序进行定期软件干预，这会增加音频流的延迟。此外，具有和不具有分散/聚集DMA功能的音频设备都可以表示为WaveRT滤波器。有关更多信息，请参见[用于实时音频流](https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/WaveRTport.doc)的[A Wave端口驱动程序](https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/WaveRTport.doc)白皮书。

**WaveRT滤波器**

WaveRT滤波器被实现为端口/微型驱动器对。在Windows Vista和更高版本中，WaveRT过滤器工厂按以下方式创建WaveRT过滤器：

* 它实例化WaveRT微型端口驱动程序对象。
* 通过使用GUID值**CLSID\_PortWaveRT**调用**[PcNewPort](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)**实例化WaveRT端口驱动程序对象。
* 它调用端口驱动程序的**[IPort :: Init](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iport-init)**方法将微型端口驱动程序绑定到端口驱动程序。

[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。端口和微型端口驱动程序通过其[IPortWaveRT](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iportwavert)和[IMiniportWaveRT](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportwavert)接口相互通信。

有关更多信息，请参见[用于实时音频流](https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/WaveRTport.doc)的[A Wave端口驱动程序](https://download.microsoft.com/download/9/c/5/9c5b2167-8017-4bae-9fde-d599bac8184a/WaveRTport.doc)白皮书。

**Windows早期版本的信息**

**Windows早期版本的WaveCyclic信息**

WaveCyclic滤波器可以表示连接到系统总线（例如ISA，PCI，PCI Express或PCMCIA）的音频设备。顾名思义，“ WavePci”表示，WavePci过滤器通常表示连接到PCI总线的设备，尽管从原理上讲，例如，WavePci设备也可以连接到ISA总线。与WaveCyclic支持的简单设备不同，WavePci支持的设备必须具有分散/聚集DMA功能。驻留在PCI总线上但没有分散/聚集DMA的音频设备可以表示为WaveCyclic滤波器，而不能表示为WavePci滤波器。

**Windows早期版本的WavePci信息**

WavePci设备能够执行往返于缓冲区的分散/聚集DMA传输，缓冲区可以位于任意内存地址，并且以任意字节对齐方式开始和结束。相反，用于WaveCyclic设备的DMA硬件仅需要将数据移入或移出设备的微型端口驱动程序分配的单个缓冲区的能力。WaveCyclic微型端口驱动程序可以自由分配满足其DMA通道有限功能的循环缓冲区。例如，典型的WaveCyclic设备的DMA通道可能需要满足以下限制的缓冲区：

* 缓冲区位于物理地址空间的特定区域中。
* 缓冲区在物理地址空间和虚拟地址空间中都是连续的。
* 缓冲区的开始和结束甚至在四字节或八字节边界上。

作为这种简单性的回报，WaveCyclic设备必须依赖于往来于循环缓冲区的数据的软件复制，而WavePci设备依赖于其DMA硬件的分散/聚集功能来避免这种复制。将波形音频数据传递到渲染设备或从捕获设备检索数据的IRP带有数据缓冲区，这些缓冲区中的每一个都包含一部分正在渲染或捕获的音频流。WavePci设备能够通过其分散/聚集DMA引擎直接访问这些缓冲区，而WaveCyclic设备则需要将数据从IRP复制到其循环缓冲区，反之亦然。

**WavePci过滤器**

**注意：Windows早期版本的WavePci信息**

WavePci滤波器被实现为端口/微型驱动器对。WavePci过滤器工厂将如下创建WavePci过滤器：

* 它实例化WavePci微型端口驱动程序对象。
* 它通过使用GUID值**CLSID\_PortWavePci**调用**[PcNewPort](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)**实例化WavePci端口驱动程序对象。
* 它调用端口驱动程序的**[IPort :: Init](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iport-init)**方法将微型端口驱动程序绑定到端口驱动程序。

[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。端口和微型端口驱动程序通过其[IPortWavePci](https://docs.microsoft.com/previous-versions/windows/hardware/drivers/ff536905(v=vs.85))和[IMiniportWavePci](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportwavepci)接口相互通信。

有关更多信息，请参见[WavePci设备的实现问题](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/implementation-issues-for-wavepci-devices)。

**WaveCyclic滤波器**

**注意：Windows早期版本的WaveCyclic信息**

WaveCyclic滤波器被实现为端口/微型驱动器对。WaveCyclic过滤器工厂按以下方式创建WaveCyclic过滤器：

* 它实例化WaveCyclic微型端口驱动程序对象。
* 它通过使用GUID值**CLSID\_PortWaveCyclic**调用**[PcNewPort来](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)**实例化WaveCyclic端口驱动程序对象。
* 它调用端口驱动程序的**[IPort :: Init](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iport-init)**方法将微型端口驱动程序绑定到端口驱动程序。

[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。端口和微型端口驱动程序通过其[IPortWaveCyclic](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iportwavecyclic)和[IMiniportWaveCyclic](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportwavecyclic)接口相互通信。

WaveCyclic过滤器的循环缓冲区始终由虚拟内存的连续块组成。**[IDmaChannel :: AllocateBuffer](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-idmachannel-allocatebuffer)**方法的端口驱动程序实现始终分配一个在物理和虚拟内存地址空间中都是连续的缓冲区。如前所述，如果WaveCyclic设备的DMA引擎在缓冲存储器上施加了其他约束，则微型端口驱动程序可以自由实现自己的缓冲区分配方法来满足这些约束。

如果操作系统拒绝原始请求，则需要准备一个较大缓冲区（例如，八个物理上连续的内存页）的WaveCyclic微型端口驱动程序以适应较小的缓冲区大小。有时可能会卸下并重新加载音频设备以重新平衡系统资源（请参阅[停止设备重新平衡资源](https://docs.microsoft.com/windows-hardware/drivers/kernel/stopping-a-device-to-rebalance-resources)）。

具有内置的总线主控DMA硬件的WaveCyclic设备称为*主设备*。或者，WaveCyclic设备可以是不具有内置DMA硬件功能的*从属设备*。从属设备必须依靠系统DMA控制器来执行其所需的任何数据传输。有关主设备和从属设备的更多信息，请参见[IDmaChannel](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-idmachannel)和[IDmaChannelSlave](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-idmachannelslave)。

WaveCyclic微型端口驱动程序可以实现自己的DMA通道对象，而不使用默认的DMA通道对象，该对象是端口驱动程序的New *Xxx* DmaChannel方法之一创建的：

[**IPortWaveCyclic :: NewMasterDmaChannel**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iportwavecyclic-newmasterdmachannel)

[**IPortWaveCyclic :: NewSlaveDmaChannel**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iportwavecyclic-newslavedmachannel)

适配器驱动程序的自定义[IDmaChannel](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-idmachannel)实现可以执行数据的自定义处理，以满足特殊的硬件约束。例如，Windows Multimedia函数使用wave格式，其中16位样本始终是带符号的值，但是音频渲染硬件可能设计为使用无符号的16位值。在这种情况下，可以编写驱动程序的自定义**[IDmaChannel :: CopyTo](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-idmachannel-copyto)**方法，以将带符号的源值转换为硬件所需的无符号的目标值。尽管此技术对于解决硬件设计缺陷很有用，但也可能导致软件开销巨大。

有关实现自己的DMA通道对象的驱动程序的示例，请参见WDK中的Sb16示例音频适配器。如果将常量OVERRIDE\_DMA\_CHANNEL定义为**TRUE**，则源代码中的条件编译语句将启用专有[IDmaChannel](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-idmachannel)对象的实现，驱动程序将使用该IDmaChannel对象代替IPortWaveCyclic :: New *Xxx* DmaChannel调用中的默认IDmaChannel对象。

# MIDI和DirectMusic滤波器

* 2017/04/20
* 6分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/midi-and-directmusic-filters.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/midi-and-directmusic-filters.md" \o "1个贡献者)

MIDI和DirectMusic过滤器代表合成，输出或捕获MIDI音乐数据的设备。应用程序通常通过DirectMusic API或Microsoft Windows Multimedia **midiOut**Xxx和**midiIn**Xxx函数来访问这些设备的功能。有关这些接口的更多信息，请参见Microsoft Windows SDK文档。

MIDI或DirectMusic 合成器过滤器接收包含时间戳MIDI事件的MIDI流作为输入。过滤器输出以下之一：

* 波形数字音频流
* 可以驱动一组扬声器的模拟音频信号

MIDI或DirectMusic 输出过滤器接收包含时间戳MIDI事件的MIDI流作为输入。过滤器将原始MIDI消息输出到外部MIDI声音模块。

MIDI或DirectMusic 捕获过滤器将从MIDI键盘或其他外部MIDI设备接收一系列原始MIDI消息作为输入。过滤器输出由带时间戳的MIDI事件组成的MIDI流。

单个MIDI或DirectMusic滤波器可以执行三个功能的组合（合成，输出和捕获），具体取决于滤波器代表的设备的功能。例如，一个纯MPU-401设备执行输出和捕获，但不执行合成。

### MIDI过滤器

MIDI滤波器被实现为端口/微型驱动器对。MIDI过滤器工厂创建MIDI过滤器，如下所示：

* 它实例化MIDI微型端口驱动程序对象。
* 它通过使用GUID值**CLSID\_PortMidi**调用[**PcNewPort来**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)实例化MIDI端口驱动程序对象。
* 它调用端口驱动程序的[**IPort :: Init**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iport-init)方法将微型端口驱动程序绑定到端口驱动程序。

[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。端口和微型端口驱动程序通过其[IPortMidi](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iportmidi)和[IMiniportMidi](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportmidi)接口相互通信。

为了支持MIDI输出和合成器设备，MIDI端口驱动程序包含一个软件音序器，该软件音序器以1毫秒的计时器分辨率将原始MIDI消息输出到微型端口驱动器。

### DirectMusic滤镜

DirectMusic过滤器提供MIDI过滤器功能的超集。超集包括以下附加功能：

* DLS（可下载声音）资源，包含描述MIDI乐器的波形和清晰度数据。一个[**KSPROPERTY\_SYNTH\_DLS\_DOWNLOAD**](https://docs.microsoft.com/previous-versions/ff537396(v=vs.85))设置属性请求下载一个DLS资源的过滤器。
* 通道组，用于扩展可选乐器的数量。该**[DMUS\_KERNEL\_EVENT](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/ns-dmusicks-_dmus_kernel_event)**结构，其用于在一个MIDI流打包每个时间标记的MIDI消息，指定要使用该消息，该消息的信道组。
* 具有100纳秒分辨率的64位时间戳，支持硬件MIDI排序。DMUS\_KERNEL\_EVENT结构指定MIDI消息的高分辨率时间戳。

使用通道组时，可以同时播放的音符数量不再限于原始MIDI规范的16个通道。它仅受合成器中可用声音的数量的限制。

DirectMusic过滤器实现为端口/微型驱动器对。DirectMusic过滤器工厂按以下方式创建DirectMusic过滤器：

* 它实例化DMus（DirectMusic）微型端口驱动程序对象。
* 通过使用GUID值**CLSID\_PortDMus**调用[**PcNewPort**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)实例化DMus端口驱动程序对象。
* 它调用端口驱动程序的[**IPort :: Init**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iport-init)方法将微型端口驱动程序绑定到端口驱动程序。

[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。端口和微型端口驱动程序通过其[IPortDMus](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/nn-dmusicks-iportdmus)和[IMiniportDMus](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/nn-dmusicks-iminiportdmus)接口相互通信。

为了支持DirectMusic合成器设备，DMus端口驱动程序包含一个低分辨率（一毫秒）软件定序器，可以在计划播放它们之前将带有时间戳的MIDI事件输出到硬件定序器的缓冲区。为了支持DirectMusic输出设备，还可以将端口驱动程序的软件音序器配置为在播放原始MIDI消息时输出它们。

### 枚举MIDI和DirectMusic设备

通过Windows Multimedia **midiInXxx**或**midiOutXxx**函数枚举MIDI输入或输出设备时，应用程序只能看到其微型端口驱动程序暴露MIDI引脚的 WDM设备。这些是用于管理原始MIDI流的引脚，但是不支持DLS和通道组等高级功能。但是，通过DirectMusic枚举设备时，应用程序可以看到同时具有MIDI引脚和DirectMusic引脚的 WDM设备。DirectMusic引脚管理带时间戳的MIDI流，并支持DLS和通道组。

实施自定义微型端口驱动程序时，硬件供应商通常会编写MIDI微型端口驱动程序或DMus微型端口驱动程序，但不能同时编写两者。MIDI微型端口驱动程序只能暴露MIDI引脚。但是，DMus微型端口驱动程序可以同时暴露MIDI和DirectMusic引脚，从而无需编写单独的MIDI微型端口驱动程序。有关DirectMusic滤波器上MIDI引脚的示例，请参阅Windows驱动程序工具包（WDK）中的Dmusuart示例音频驱动程序。

为MIDI或DirectMusic引脚指定数据范围时，MIDI或DMus微型端口驱动程序为MIDI引脚指定KSDATAFORMAT\_TYPE\_MUSIC类型的主要格式，为MIDI引脚指定KSDATARANGE\_SUBTYPE\_MIDI类型的子格式，或者为DirectMusic引脚指定KSDATARANGE\_SUBTYPE\_DIRECTMUSIC。MIDI和DirectMusic引脚的数据范围描述符的示例分别出现在[MIDI流数据范围](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/midi-stream-data-range)和[DirectMusic流数据范围中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/directmusic-stream-data-range)。

MIDI过滤器上的MIDI引脚实例公开[IMiniportMidiStream](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportmidistream)接口。DirectMusic过滤器上的MIDI或DirectMusic引脚实例公开了[IMXF](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/nn-dmusicks-imxf)接口。

在Windows Me / 98中，DirectMusic有时枚举同一MPU-401设备两次。原因是某些硬件供应商将其MPU-401设备同时作为传统的WDM之前的MIDI设备和WDM设备公开。对于旧式设备，DirectMusic枚举了一个MPU-401设备，该设备表示从DMusic.dll到Ihvaudio.dll的直接路径。对于WDM设备，DirectMusic通过a回路径枚举同一MPU-401设备，该path回路径包含以下组件序列：

1. DMusic.dll
2. DMusic16.dll
3. MMSystem.dll
4. WDMAud.drv
5. WDMAud.sys
6. 供应商的微型端口驱动程序

Windows多媒体控制面板（Mmsys.cpl）中显示的MIDI合成器将具有与WDM设备相同的名称。

### 系统提供的端口和微型端口驱动程序

PortCls系统驱动程序内置了一些系统提供的MIDI和DMus微型端口驱动程序：

* FMSynth微型端口驱动程序为实现OPL3风格FM合成的MIDI设备提供接口。
* UART微型端口驱动程序支持带有MPU-401硬件接口的MIDI设备，但是该驱动程序现在已过时（在Windows 98 Gold之后），并且仅受现有适配器驱动程序支持。新的适配器驱动程序代码应改为使用DMusUART微型端口驱动程序（在Windows 98 SE和Windows Me中以及在Windows 2000及更高版本中），该驱动程序将替代UART并实现其功能的超集。

适配器驱动程序可以通过调用[**PcNewMiniport**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewminiport)函数来访问系统提供的微型端口驱动程序。Windows驱动程序工具包（WDK）中还包括FMSynth和DMusUART微型端口驱动程序作为示例音频驱动程序。通过修改这些示例中的源代码，硬件供应商可以扩展驱动程序以管理其设备的专有功能。

DMusUART是DMus微型端口驱动程序的一个示例，该驱动程序公开了MIDI和DirectMusic引脚，但不支持DLS下载或硬件排序。微型端口驱动程序的DirectMusic呈现引脚具有一个合成节点（[**KSNODETYPE\_SYNTHESIZER**](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-synthesizer)），该节点支持多个[KSPROPSETID\_Synth](https://docs.microsoft.com/windows-hardware/drivers/audio/kspropsetid-synth)属性。微型端口驱动程序将其自身包含在类别KSCATEGORY\_RENDER和KSCATEGORY\_CAPTURE中，但不包含在KSCATEGORY\_SYNTHESIZER中（因为它不包含内部合成器）。有关详细信息，请参见WDK中的DMusUART示例音频驱动程序。

请注意，在Windows XP和更高版本中，MIDI和DMus端口驱动程序使用相同的内部软件实现。这意味着在调用**PcNewPort**时，**CLSID\_PortMidi**和**CLSID\_PortDMus** GUID是等效的。为Windows的早期版本编写的应用程序应该不会由于合并MIDI和DMus端口驱动程序而导致行为发生任何变化。

# 拓扑过滤器

* 2017/04/20
* 7分钟阅读
  + [[https://github.com/DOMARS.png?size=32](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/topology-filters.md)](https://github.com/MicrosoftDocs/windows-driver-docs/blob/staging/windows-driver-docs-pr/audio/topology-filters.md" \o "1个贡献者)

甲拓扑过滤器表示电路的一个处理的各种波和MIDI流被在卡上管理之间的交互的音频适配卡上的部分。该电路进行渲染流的混合和捕获流的多路复用。

拓扑过滤器提供桥接引脚（请参阅[音频过滤器图表](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filter-graphs)），这些桥接引脚表示音频适配器与外部设备的物理连接。这些连接通常承载驱动扬声器的模拟输出信号和来自麦克风的模拟输入信号。拓扑滤波器的桥接引脚也可能代表模拟输入和输出插孔，甚至可能代表数字输入和输出连接器。

术语“拓扑过滤器”在某种意义上是误称。尽管其名称，拓扑过滤器只是公开其内部拓扑或布局的几种音频过滤器中的一种。尽管拓扑过滤器包含关键的拓扑功能，但不一定包含适配器的整个拓扑。Wave和MIDI滤波器具有自己的拓扑。例如，最小的WaveCyclic或WavePci滤波器（请参阅[Wave滤波器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/wave-filters)）可能会暴露由两个引脚以及DAC（数模转换器）或ADC（模数转换器）组成的拓扑，具体取决于底层设备执行音频渲染或捕获。

拓扑过滤器实现为端口/微型对。拓扑过滤器工厂创建拓扑过滤器，如下所示：

* 它实例化拓扑微型端口驱动程序对象。
* 通过使用GUID值**CLSID\_PortTopology**调用[**PcNewPort**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcnewport)实例化拓扑端口驱动程序对象。
* 它调用端口驱动程序的[**IPort :: Init**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iport-init)方法将微型端口驱动程序绑定到端口驱动程序。

[子设备创建中](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/subdevice-creation)的代码示例说明了此过程。

拓扑端口和微型端口驱动程序通过它们各自的[IPortTopology](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iporttopology)和[IMiniportTopology](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiporttopology)接口相互通信。与wave和MIDI端口以及微型端口驱动程序相比，这些接口相对简单，因为拓扑过滤器不需要显式管理通过其引脚的流。拓扑过滤器的引脚表示适配器硬件中的硬连线连接。拓扑滤波器引脚下方的物理连接通常会承载模拟音频信号，但可能会承载数字音频流，具体取决于硬件实现。

与[IMiniportWaveCyclic](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportwavecyclic)，[IMiniportWavePci](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportwavepci)，[IMiniportMidi](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiportmidi)和[IMiniportDMus](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/nn-dmusicks-iminiportdmus)接口[相反](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/dmusicks/nn-dmusicks-iminiportdmus)，[IMiniportTopology](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nn-portcls-iminiporttopology)接口没有**NewStream**方法。

拓扑过滤器的大多数功能由其属性处理程序提供。拓扑筛选器主要用于向[SysAudio系统驱动程序](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/kernel-mode-wdm-audio-components" \l "sysaudio_system_driver)和使用Microsoft Windows Multimedia Mixer API的应用程序提供拓扑信息。拓扑过滤器中的属性处理程序提供对音频适配器通常提供的各种控件（例如音量，均衡和混响）的访问。通过属性请求，混合器API可以枚举适配器硬件中的控制节点，发现节点之间的连接，以及查询和设置节点的控制参数。SndVol32应用程序（请参阅[SysTray和SndVol32](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/systray-and-sndvol32)）使用混合器API查找适配器的每流音量和静音控件。

生成过滤器图形时，SysAudio在拓扑过滤器的引脚上查询[**KSPROPERTY\_PIN\_PHYSICALCONNECTION**](https://docs.microsoft.com/windows-hardware/drivers/stream/ksproperty-pin-physicalconnection)属性，以确定哪个wave，MIDI或DirectMusic过滤器引脚连接到哪个拓扑过滤器引脚。

与Wave，MIDI或DirectMusic滤波器不同，拓扑滤波器不会实例化引脚。因此，没有引脚对象可用于处理对拓扑过滤器的引脚属性的查询。拓扑过滤器本身会处理有关其引脚上的物理连接的所有查询。有关更多信息，请参见[KSPROPSETID\_Pin](https://docs.microsoft.com/windows-hardware/drivers/stream/kspropsetid-pin)。

与其他类型的音频过滤器类似，拓扑过滤器使用[**PCCONNECTION\_DESCRIPTOR**](https://docs.microsoft.com/previous-versions/windows/hardware/drivers/ff537688(v=vs.85))结构数组来描述其内部拓扑。微型端口驱动程序在从**[IMiniport :: GetDescription](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-iminiport-getdescription)**方法输出的[**PCFILTER\_DESCRIPTOR**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/ns-portcls-pcfilter_descriptor)结构中公开此数组。该数组将拓扑指定为拓扑过滤器的节点和引脚之间的连接列表（请参阅[Nodes and Connections](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/nodes-and-connections)）。所述[WDMAud系统驱动](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/user-mode-wdm-audio-components" \l "wdmaud_system_driver)平移这些连接和节点到混合器线，并控制该混合器API暴露给应用程序。如[音频过滤器中所述](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filters)，KS滤波器上的输入引脚映射到SRC混频器线，而滤波器上的输出引脚映射到DST混频器线。

典型的音频适配器可以通过扬声器播放wave和MIDI文件，并且可以捕获来自麦克风和MIDI合成器的音频信号。下面的代码示例包含用于拓扑过滤器的PCCONNECTION\_DESCRIPTOR数组，它公开了这些功能：

C ++复制

// topology pins

enum

{

KSPIN\_TOPO\_WAVEOUT\_SRC = 0,

KSPIN\_TOPO\_SYNTHOUT\_SRC,

KSPIN\_TOPO\_SYNTHIN\_SRC,

KSPIN\_TOPO\_MIC\_SRC,

KSPIN\_TOPO\_LINEOUT\_DST,

KSPIN\_TOPO\_WAVEIN\_DST

};

// topology nodes

enum

{

KSNODE\_TOPO\_WAVEOUT\_VOLUME = 0,

KSNODE\_TOPO\_WAVEOUT\_MUTE,

KSNODE\_TOPO\_SYNTHOUT\_VOLUME,

KSNODE\_TOPO\_SYNTHOUT\_MUTE,

KSNODE\_TOPO\_MIC\_VOLUME,

KSNODE\_TOPO\_SYNTHIN\_VOLUME,

KSNODE\_TOPO\_LINEOUT\_MIX,

KSNODE\_TOPO\_LINEOUT\_VOLUME,

KSNODE\_TOPO\_WAVEIN\_MUX

};

static PCCONNECTION\_DESCRIPTOR MiniportConnections[] =

{

// FromNode---------------------FromPin------------------ToNode-----------------------ToPin

{ PCFILTER\_NODE, KSPIN\_TOPO\_WAVEOUT\_SRC, KSNODE\_TOPO\_WAVEOUT\_VOLUME, 1 },

{ KSNODE\_TOPO\_WAVEOUT\_VOLUME, 0, KSNODE\_TOPO\_WAVEOUT\_MUTE, 1 },

{ KSNODE\_TOPO\_WAVEOUT\_MUTE, 0, KSNODE\_TOPO\_LINEOUT\_MIX, 1 },

{ PCFILTER\_NODE, KSPIN\_TOPO\_SYNTHOUT\_SRC, KSNODE\_TOPO\_SYNTHOUT\_VOLUME, 1 },

{ KSNODE\_TOPO\_SYNTHOUT\_VOLUME, 0, KSNODE\_TOPO\_SYNTHOUT\_MUTE, 1 },

{ KSNODE\_TOPO\_SYNTHOUT\_MUTE, 0, KSNODE\_TOPO\_LINEOUT\_MIX, 2 },

{ PCFILTER\_NODE, KSPIN\_TOPO\_SYNTHIN\_SRC, KSNODE\_TOPO\_SYNTHIN\_VOLUME, 1 },

{ KSNODE\_TOPO\_SYNTHIN\_VOLUME, 0, KSNODE\_TOPO\_WAVEIN\_MUX, 1 },

{ PCFILTER\_NODE, KSPIN\_TOPO\_MIC\_SRC, KSNODE\_TOPO\_MIC\_VOLUME, 1 },

{ KSNODE\_TOPO\_MIC\_VOLUME, 0, KSNODE\_TOPO\_WAVEIN\_MUX, 2 },

{ KSNODE\_TOPO\_LINEOUT\_MIX, 0, KSNODE\_TOPO\_LINEOUT\_VOLUME, 1 },

{ KSNODE\_TOPO\_LINEOUT\_VOLUME, 0, PCFILTER\_NODE, KSPIN\_TOPO\_LINEOUT\_DST },

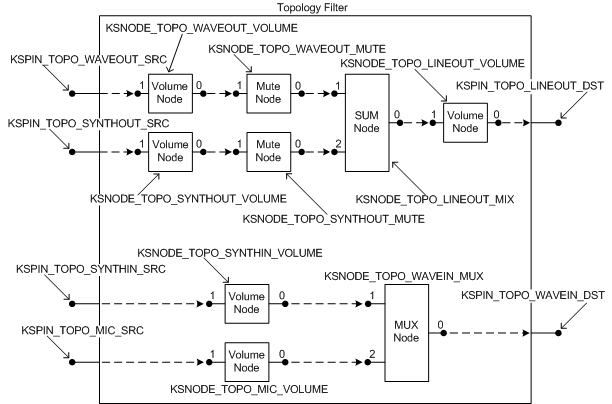
{ KSNODE\_TOPO\_WAVEIN\_MUX, 0, PCFILTER\_NODE, KSPIN\_TOPO\_WAVEIN\_DST }

};

前面的代码示例中的常量[**PCFILTER\_NODE**](https://docs.microsoft.com/previous-versions/ff537695(v=vs.85))是空节点ID，并且在头文件Portcls.h中定义。有关如何使用此常量来区分过滤器上的外部引脚和节点上的逻辑引脚的说明，请参见[**PCCONNECTION\_DESCRIPTOR**](https://docs.microsoft.com/previous-versions/windows/hardware/drivers/ff537688(v=vs.85))。

前面的代码示例中的每个引脚名称都以“ SRC”或“ DST”结尾，具体取决于混合器API是否将引脚映射到源或目标混合器行。为避免混淆，请记住，源和目标混频器线分别映射到接收（输入）和源（输出）KS滤波器引脚。有关更多信息，请参见[音频过滤器](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-filters)。

前面的代码示例中的PCCONNECTION\_DESCRIPTOR数组在下图中描述了拓扑过滤器。



图中的拓扑过滤器在左侧有四个输入（接收）引脚，在右侧有两个输出（源）引脚。连接顶部两个输入引脚和顶部输出引脚的数据路径混合了从正在播放的wave和MIDI流中渲染的两个模拟信号。连接底部两个输入引脚和底部输出引脚的数据路径对捕获的正在记录的模拟信号进行多路复用。

四个输入引脚的操作如下：

* KSPIN\_TOPO\_WAVEOUT\_SRC引脚物理连接到滤波器的输出引脚，该滤波器输出来自.wav文件等源的波流，以在该引脚上产生模拟信号。
* KSPIN\_TOPO\_SYNTHOUT\_SRC引脚物理连接到合成器滤波器的输出引脚，合成器滤波器可能会渲染例如来自.mid文件之类的源的MIDI流，以在该引脚上产生模拟信号。
* KSPIN\_TOPO\_SYNTHIN\_SRC引脚物理连接到生成模拟信号的合成器。（请注意，更实用的硬件设计可能会从MPU-401 MIDI接口获取MIDI输入流，并将其直接转换为波形格式，从而完全绕开拓扑过滤器。）
* KSPIN\_TOPO\_MIC\_SRC引脚物理连接到输入插孔，该输入插孔接收来自麦克风的模拟信号。

两个输出引脚的操作如下：

* KSPIN\_TOPO\_LINEOUT\_DST引脚物理连接到模拟线路输出插孔，该插孔通常驱动一组扬声器。
* KSPIN\_TOPO\_WAVEIN\_DST引脚物理连接到波形滤波器的输入引脚，后者将模拟信号转换为波形流并将其写入目标文件（例如.wav文件）。

音量和静音节点（请参阅[**KSNODETYPE\_VOLUME**](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-volume)和**[KSNODETYPE\_MUTE](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-mute)**）用于控制各种流的音量级别。SUM节点（请参阅[**KSNODETYPE\_SUM**](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-sum)）混合来自wave和MIDI输入的音频流。MUX节点（请参阅[**KSNODETYPE\_MUX**](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-mux)）在两个输入流之间进行选择。

该图使用虚线箭头表示两个节点之间或引脚与节点之间的连接。箭头指向数据流的方向。该图显示了总共13个连接，每个连接对应于前面的代码示例中PCCONNECTION\_DESCRIPTOR数组中的13个元素之一。

除了拓扑过滤器之外，适配器驱动程序还会创建其他过滤器（波形，FM合成器，波形表等），它们连接到拓扑过滤器上的引脚。

例如，物理连接到拓扑滤波器的KSPIN\_TOPO\_WAVEOUT\_SRC引脚的滤波器包含一个DAC（由**[KSNODETYPE\_DAC](https://docs.microsoft.com/windows-hardware/drivers/audio/ksnodetype-dac)**节点表示），该DAC 将PCM数据转换为模拟信号，并输出到拓扑滤波器的引脚。物理连接到拓扑滤波器的KSPIN\_TOPO\_SYNTHOUT\_SRC引脚的FM合成器或波表合成器滤波器将MIDI数据类似地转换为模拟信号，然后将其输出到拓扑滤波器的引脚。拓扑滤波器混合来自这两个引脚的模拟信号，并将混合后的信号输出到扬声器。

需要将拓扑过滤器与代表同一适配器卡上其他硬件设备的其他过滤器的物理连接与其他类型的过滤器连接区分开。例如，可以在软件控制下连接或断开wave，MIDI和DirectMusic滤波器上的某些引脚。

在设备启动期间，适配器驱动程序通过每个连接调用一次[**PcRegisterPhysicalConnection来**](https://docs.microsoft.com/windows-hardware/drivers/ddi/content/portcls/nf-portcls-pcregisterphysicalconnection)注册拓扑过滤器的物理连接。端口驱动程序需要此信息以响应[**KSPROPERTY\_PIN\_PHYSICALCONNECTION**](https://docs.microsoft.com/windows-hardware/drivers/stream/ksproperty-pin-physicalconnection)获取属性请求。