**Windows APO(Audio Processing Objects, 音频处理对象)**

原文时间是:2017/04/20, 适用对象是Windows 10

APO最初是在Windows Vista中引入的，您可能会看到对早期系统APO-sAPO的引用。 有关详细信息，请参阅Windows Vista中的自定义音频效果([Custom Audio Effects in Windows Vista](https://go.microsoft.com/fwlink/p/?linkid=106024))。 本白皮书可能会参考较旧的[COM](https://go.microsoft.com/fwlink/p/?linkid=106293) 和 [UI开发](https://go.microsoft.com/fwlink/p/?linkid=106294)主题。

# 1. APO架构

音频处理对象（APO）为Windows音频流提供了可定制化的软件数字信号处理.

## 1.1 APO概览

Windows允许OEM和第三方音频硬件制造商将自定义数字信号处理效果作为其音频驱动程序增值功能的一部分。 这些效果打包为用户模式系统效果音频处理对象（APO）。

音频处理对象（APO）为Windows音频流提供基于软件的数字信号处理。 APO是COM主机对象(host object)，包含为提供特定数字信号处理（DSP）效果而编写的算法。 这种能力被非正式地称为“音效”。 APO的示例包括均衡器，混响，颤音，回声消除（AEC）和自动增益控制（AGC）。 APO是基于COM的实时进程中对象。

注意: 本文档中的描述和术语主要涉及输出设备。 但是，该技术是对称的，并且对于输入设备基本上是反向的。

### 1.1.1 软件APOs vs. 硬件 DSP

硬件数字信号处理器（DSP）是专用微处理器（或SIP块），其架构针对数字信号处理的操作需求进行了优化。与使用软件APO相比，在专用硬件中实现音频处理可能具有显著的优势。一个优点是使用硬件实现的DSP可以降低CPU使用和相关的功耗。

还有其他优点和缺点需要考虑，具体到在实施基于软件的APO之前需要考虑的项目目标和约束。

基于软件的音效在流初始化时插入软件设备管道中。这些解决方案可以在主CPU上执行所有效果处理，而不依赖于外部硬件。当驱动程序和硬件仅支持RAW处理时，此类解决方案最适用于传统的Windows音频解决方案，如HDAudio，USB和蓝牙设备。有关RAW处理的更多信息，请参阅音频信号处理模式([Audio Signal Processing Modes](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-signal-processing-modes))。

### 1.1.2 硬件DSP的代理APO(Proxy APO)

在硬件DSP中应用的任何效果都需要通过代理APO进行通告。 Microsoft提供默认代理APO（MsApoFxProxy.dll）。要使用Microsoft提供的APO，必须支持此属性集和属性。

* [KSPROPSETID\_AudioEffectsDiscovery](https://docs.microsoft.com/windows-hardware/drivers/audio/kspropsetid-audioeffectsdiscovery)
* [KSPROPERTY\_AUDIOEFFECTSDISCOVERY\_EFFECTSLIST](https://docs.microsoft.com/previous-versions/windows/hardware/drivers/dn457706(v=vs.85))

（可选）您可以实现自己的代理APO。

### 1.1.3 Windows 提供的(系统) APOs

Windows安装一组默认的APO，它们提供许多不同的音频效果。有关系统提供的APO效果的列表，请参阅音频信号处理模式([Audio Signal Processing Modes](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-signal-processing-modes)).

OEM可以包括所有系统提供的APO，或者使用自定义APO替换其中的部分或全部。

### 1.1.4 自定义APOs

可以通过添加其他音频效果来创建自定义APO以增强Windows音频体验。

OEM可以在发布Windows时包含所提供的Windows APO和自定义APO的任意组合。

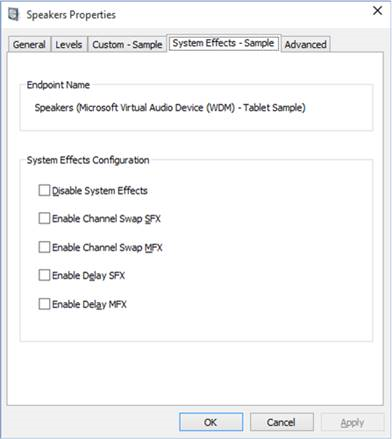
OEM或第三方可以安装自定义APO，以在购买设备后增强音频体验。当用户使用标准INF文件安装音频设备驱动程序时，他们可以自动访问系统的APO。独立硬件供应商（IHV）和原始设备制造商（OEM）可以提供额外的自定义系统效果，同时仍使用Microsoft类驱动程序(class driver))。他们通过将DSP算法打包为APO并修改标准INF文件以将其APO插入到音频引擎的信号处理图中来实现。

有关创建自定义APO的详细信息，请参阅下一部分 2.实现音频处理对象(Implementing Audio Processing Objects).

### 1.1.5 定制APO属性页面

在台式PC上，可以创建自定义APO属性页面，以允许用户配置与自定义APO关联的设置。 此自定义APO页面可作为标准Windows音频设置的扩展提供给用户。

SYSVAD audio是一个自定义APO控件示例程序。 以下屏幕截图显示了SYSVAD audio 的Swap APO示例程序的属性页。



有关添加APO对话框面板的更多信息，请参阅第三部分 3. 实现用于配置APO效果的UI([Implementing a UI for Configuring APO Effects](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/implementing-a-ui-for-configuring-apo-effects)).

### 1.1.6 定制APO测试和要求

Microsoft HLK提供可与APO一起使用的测试。 有关音频测试的更多信息，请参阅[Device.Audio Testing](https://docs.microsoft.com/previous-versions/windows/hardware/hck/jj123955(v=vs.85)) 和[Device.Audio Tests](https://docs.microsoft.com/previous-versions/windows/hardware/hck/jj124726(v=vs.85)).

在使用APO时，以下两项测试特别有用。

[Verify Audio EffectsDiscovery (Manual) - Certification](https://docs.microsoft.com/previous-versions/windows/hardware/hck/dn456312(v=vs.85))

[SysFX Test](https://docs.microsoft.com/previous-versions/windows/hardware/hck/jj124017(v=vs.85))

有关支持APO的音频要求的信息，请参阅[Device.Audio要求](https://docs.microsoft.com/previous-versions/windows/hardware/cert-program/deviceaudio-requirements).

### 1.1.7 自定义APO 工具和实用程序

可以使用"Audio Effects Discovery Sample" 来研究可用的音频效果. 该例子演示了如何在渲染和捕获音频设备上查询音频效果以及如何使用监控音频效果的更改。 它包含在SDK示例中，可以使用以下链接下载()：

[Audio effects discovery sample](https://code.msdn.microsoft.com/windowsapps/Audio-effects-discovery-5fd65c15)

此例子在2019下编译产生以下错误:

1. vccorlib.h(1560): error C3946: 'T': typeid cannot be applied to this type, 修正方法为修改SuspensionManager.cpp中第450行为return (char16\_t)reader->ReadUInt16();中的char16\_t为wchar\_t
2. error APPX0501: Validation error. Unspecified error 'FxDscvry CPP' violates pattern constraint of '\bms-resource:.{1,256}'. 这个问题我没能解决, 有可参考的解决方案是: <https://www.oipapio.com/question-9250619> 待更新

### 1.1.8 应用程序如何查询音效状态

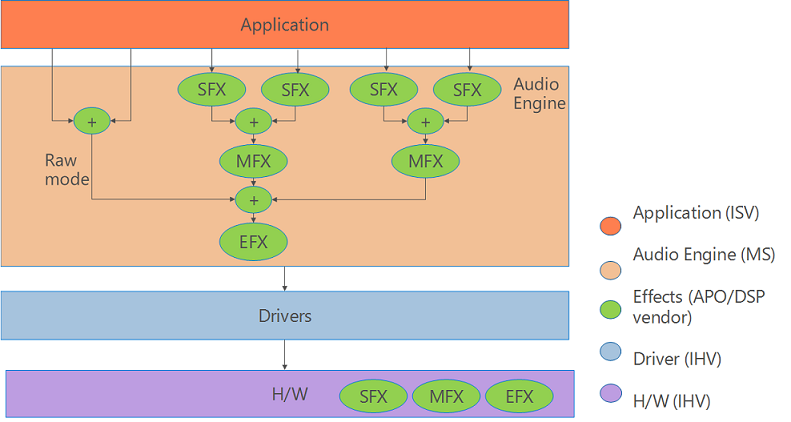
引用程序可以调用API来查询当前系统中有哪些音效正在使用. 更多信息请查看 [AudioRenderEffectsManager class](https://docs.microsoft.com/uwp/api/Windows.Media.Effects.AudioRenderEffectsManager).

## 1.2 APO架构

### 1.2.1音效的位置

以APO方式实现的音效可以放在三个位置: SFX (Stream effects, 流音效), MFX (Mode effects, 模式音效), 和 EFX (Endpoint effects, 端点音效).

下图展示了SFX,MFX和EFX的位置:



### 1.2.2 SFX (Stream Effect, 流音效)

SFX APO 为每个流实现一个音效实例. SFX的位置在mix之前(渲染设备)或tee之后(捕获设备), 而且可用于在mixer 之前改变通道数目. RAW模式下是不经过SFX的.

有些版本的Windows, 作为优化, 在raw模式下是不加载SFX或MFX APO的.

* Windows 8.1 在raw模式下不载入SFX或MFX
* Windows 10 在raw模式下载入MFX但不载入SFX

### 1.2.3 MFX (Mode Effect, 模式音效)

同模式的流会先被Mix, 进入MFX进行处理; 各MFX的输出再次进行mix; 对捕获设备则 反过来.任何场景相关的音效或者和流细节(specifics)无关的音效应该放在这里. 使用一个MFX实例支持多个同属性(码率和格式)的流效率会更高.

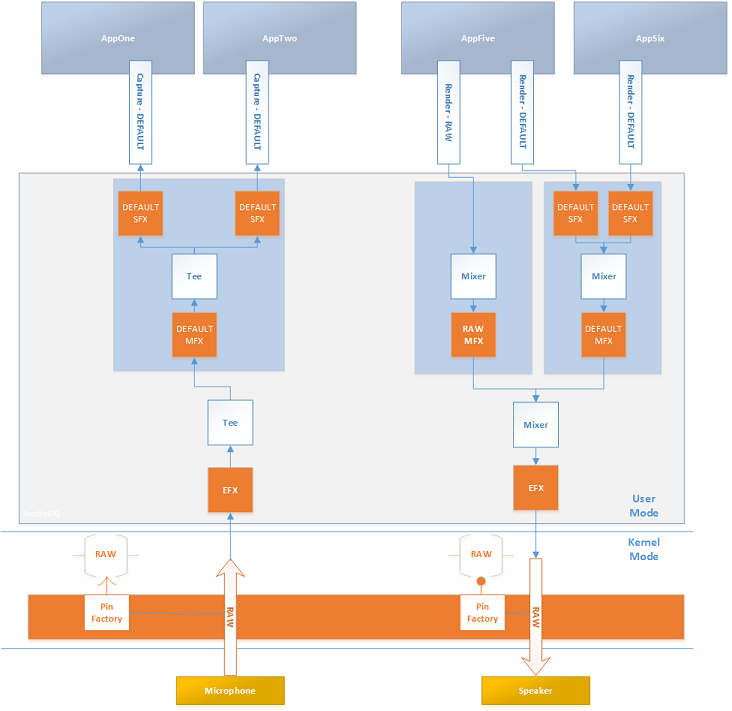
### 1.2.4 EFX(Endpoint Effect, 端点音效)

EFX对使用同一个端点的流生效. EFX总是生效, 包括RAW模式. 其位置在Mix(渲染设备)之后或Tee(捕获设备)之前. 音效放入EFX时要谨慎, 如有疑问, 则放到MFX. 扬声器保护和补偿的音效应该放到EFX.

### 1.2.5 多个定制的APO音效

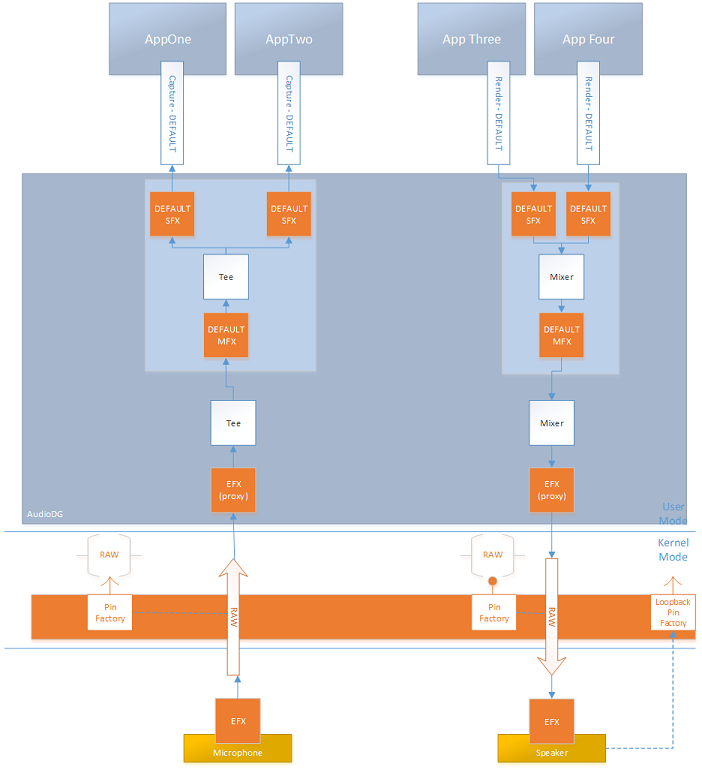
可以配置多个APO音效来支持不同的应用.

下图描述了多个应用如何访问不同的SFX, MFX, EFX APO的组合. 所有的APO都是基于COM并运行在用户模式, 此图中音效没有运行硬件和内核模式.



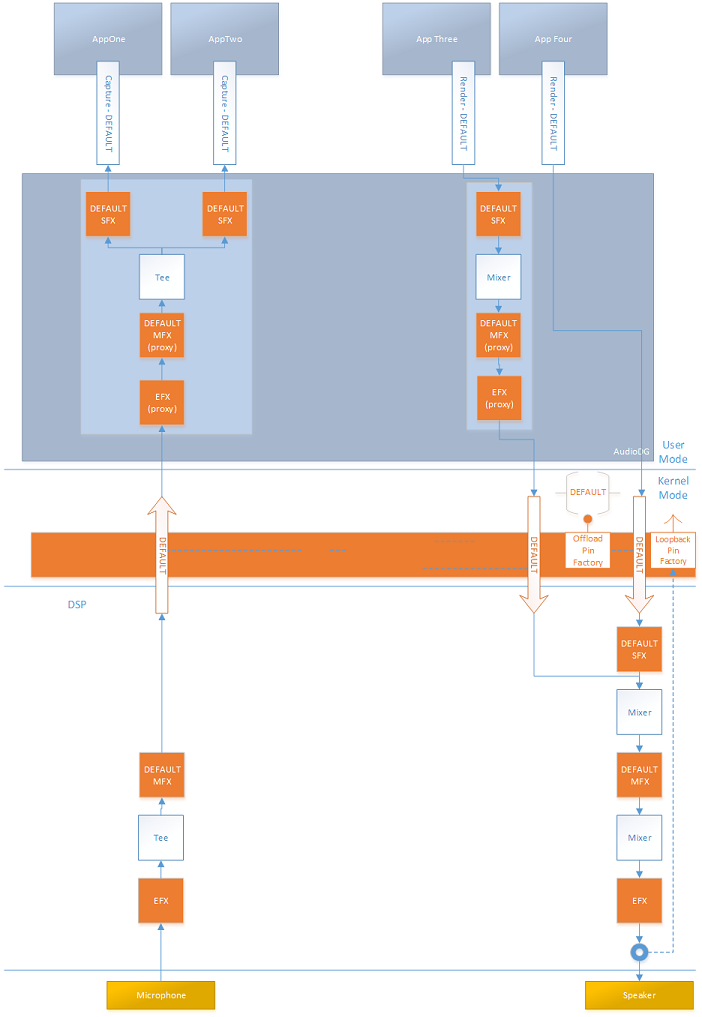
### 1.2.6 软件MFX+硬件EFX

下图展示了渲染设备和捕获设备中使用软件MFX和硬件EFX的方案.



### 1.2.7 带DSP系统上的硬件音效

下图展示了带DSP系统如何在硬件上实现音效的. 在此场景下,会实现一个代理(Proxy)APO来告知应用程序硬件里面实现了哪些音效.



# 2. 实现APO

本节描述如何实现APO, 更多信息请参考APO架构章节.

## 2.1 实现自定义的APO

自定义APO以进程内COM对象(in-process COM objects)的形式实现, 所以他们可以运行在用户模式, 并包装为动态链接库(DLL)的形式.根据我们想把他们放到信号链路图的什么位置, 有三种类型的APO.

* SFX(Stream effects, 流音效)
* MFX(Mode effects, 模式音效)
* EFX(Endpoint effects, 端点音效)

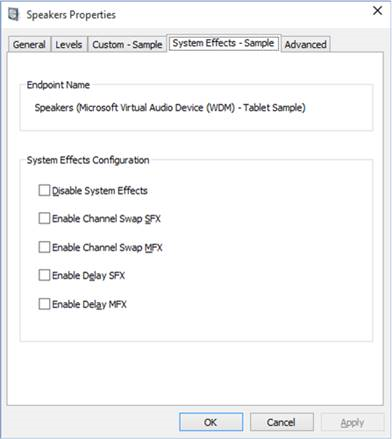
对于每种APO类型,每个逻辑设备都可以关联一个APO(在Window10 1803版本之后, 已经支持PKEY\_CompositeFX系列的, 请参考[Audio INF File Settings](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/media-class-inf-extensions)). 更多关于模式和音效的信息, 请参考音频信号处理模式([Audio Signal Processing Modes](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-signal-processing-modes)).

实现APO的时候可以从CBaseAudioProcessingObject派生出来, 相关定义在Baseaudioprocessingobject.h中. CBaseAudioProcessingObject已经实现了APO所需的许多功能的默认实现, 特别是三个必需接口, 但是不包含**[IAudioProcessingObjectRT::APOProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess)**函数, 开发者需要在此基础上实现自己的定制功能/函数.

执行以下三步可以定制自己的APO.

1. 创建自定义APO的com对象来提供期望的音频处理.
2. 可选:创建UI以配置自定义的APO.
3. 创建INF文件安装和注册APO和自定义的UI.

关于实现自定义属性也的更多消息, 参考实现APO配置UI章节. 下面的截屏图片展示了SwapAPO的属性页.



## 2.2 开发自定义APO过程中的设计考虑

所有的自定义APO必须包含以下基本特性:

* APO必须包含输入输出连接(connection), 这些连接的形式是音频buffer(注:数据数组指针), 可以是多通道的.
* APO可以通过**[IAudioProcessingObjectRT::APOProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess)**修改通过它的音频数据. APO无法修改底层逻辑设备的设置(包括其KS拓扑).
* 除了IUnknown, APO还必须支持(expose)以下接口:
  + [IAudioProcessingObject](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudioprocessingobject). 此接口处理设定(setup)任务比如初始化和格式沟通(format negotiation).
  + [IAudioProcessingObjectConfiguration](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudioprocessingobjectconfiguration). 配置(configure)接口.
  + [IAudioProcessingObjectRT](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudioprocessingobjectrt). 实时接口. 被实时处理线程调用, 执行音频处理.
  + [IAudioSystemEffects](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudiosystemeffects). 音频引擎所支持的音效接口.
* 所有的APO必须满足实时系统兼容要求, 意思是:
  + 所有实时接口必须实现为无阻塞函数. 不应阻塞, 不应使用分页内存(paged memory, 译注:使用AERT\_Allocate来申请和锁定内存, 使用AVRT\_CODE\_BEGIN等宏来配置代码区段, 更多概念请[锁定内存](https://docs.microsoft.com/en-us/windows/win32/memory/creating-guard-pages)相关内容, 或参考<深入解析Windows操作系统>第四版7.2的相关章节), 不应调用任何阻塞类的系统例程(routines).
  + APO处理的所有的buffer都必须不可调出的(nonpageable).处理路径上的代码和数据都应该是不可调出的.
  + APO不应引入明显的延迟.
* 自定义APO不应暴露(expose) [IAudioProcessingObjectVBR](https://github.com/tpn/winsdk-10/blob/master/Include/10.0.10240.0/um/audioenginebaseapo.idl)接口(译注:用来统计最大输入和输出帧数的接口,不应在实时进程中调用).

注: 关于所需接口的更多信息, 请参考Windows Kits\<build number>\Include\um 目录下的Audioenginebaseapo.h 和Audioenginebaseapo.idl文件.

## 2.3 使用例子代码来加速开发进程

使用例子程序SYSVAD Swap APO作为模板来开发可以加速自定义APO的开发进度. Swap APO这个例子被用来展示APO的一些特性. 此例子交换左右声道的数据, 实现了SFX和MFX音效. 可以通过使用属性页窗口来开关通道切换音效..

例子程序SYSVAD audio可以在Github([Windows Driver Samples GitHub](https://github.com/Microsoft/Windows-driver-samples))找到.

具体的可以通过访问以下链接查看Sysvad audio例子:

<https://github.com/Microsoft/Windows-driver-samples/tree/master/audio/sysvad>

### 2.3.1 从Github下载和解压Sysvad audio例子程序

按照以下步骤下载和打开SYSVAD例子.

a. 你可以考虑使用GitHub玩转例子. 也可以下载整个universal驱动例子的zip包.

<https://github.com/Microsoft/Windows-driver-samples/archive/master.zip>

b. 把这个master.zip文件下载到本地硬盘.

c. 右键点击*Windows-driver-samples-master.zip*,选择全部解压(Extract All**)**. 指定一个新目录或者选择一个已经存在的目录来存储这些解压文件. 比如可以指定*C:\DriverSamples\*作为新目录来解压文件.

d. 在文件解压完成后, 访问下面的子目录:

*C:\DriverSamples\Audio\Sysvad*

### 2.3.2 在Visual Studio中打开驱动的solution文件

在微软Visual Studio中, 点击文件(**File)** > 打开(**Open)** > 工程/方案(**Project/Solution...)** 来访问包含了解压文件的目录(比如, *C:\DriverSamples\Audio\Sysvad*). 双击*Sysvad.sln* 文件来打开.

在Visual Studio中找到Solution Explorer (如果还没打开, 在视图(**View**)菜单中选择**Solution Explorer**)打开此窗口. 在此窗口中, 可以看到此solution下面有好几个工程(译注:原文是6个,现在已经有14个了).

### 2.3.3. SwapAPO例子代码

在SYSVAD例子中,其中一个是最值得APO开发者关注的工程师SwapAPO.

|  |  |
| --- | --- |
| 工程 | 描述 |
| SwapAPO | 例子APO的示例代码 |

Sysvad例子中的其他工程总结如下(译注:原文没包含最新的一些工程, 后面看能不能补全).

|  |  |
| --- | --- |
| 工程 | 描述(译注:这里的代码我还没细看, 先不乱翻) |
| PhoneAudioSample | Sample code for a mobile audio driver. |
| TabletAudioSample | Sample code for an alternate audio driver. |
| KeywordDetectorAdapter | Sample code for a keyword detector adapter |
| EndpointsCommon | Sample code for common endpoints. |

SwapAPO例子的主要头文件是swapapo.h. 其他主要的代码文件总结如下.

|  |  |
| --- | --- |
| 文件 | 描述 |
| Swap.cpp | Swap APO核心实现 |
| SwapAPOMFX.cpp | CSwapAPOMFX类 |
| SwapAPOSFX.cpp | CSwapAPOSFX类 |
| SwapAPODll.cpp | DLL的导出函数(Exports) |
| SwapAPODll.idl | 接口定义和DLL的coclass |
| SwapAPOInterface.idl | Swap APO的接口和类型定义. |
| swapapodll.def | COM/Dll 导出函数的定义 |

## 2.4 实现COM对象APO

实现APO的时候可以从**CBaseAudioProcessingObject**派生出来, 相关定义在Baseaudioprocessingobject.h中. 这样做可以通过在**CBaseAudioProcessingObject** 基类的基础上增加新功能来实现自定义APO. **CBaseAudioProcessingObject**已经实现了APO所需的许多功能的默认实现, 特别是三个必需接口, 但是不包含**[IAudioProcessingObjectRT::APOProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess)**函数.

通过使用**CBaseAudioProcessingObject**, 可以很容易就实现一个APO. 如果APO没有特殊的格式要求, 并处理32位浮点数格式, **CBaseAudioProcessingObject**接口中的默认实现就应该够了. 在默认实现的基础上, 只有三个函数必须实现: [**IAudioProcessingObject::IsInputFormatSupported**](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobject-isinputformatsupported), [**IAudioProcessingObjectRT::APOProcess**](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess), 和**ValidateAndCacheConnectionInfo**.

在**CBaseAudioProcessingObject**类的基础上开发你自己的APO, 需要执行以下几步:

1. 从**CBaseAudioProcessingObject**派生一个类.

以下的C++示例代码展示了如何从**CBaseAudioProcessingObject**派生一个类. 其具体的实现请参考上一节提到的Swap例子, 参考*Swapapo.h* 文件.

// Custom APO class - LFX

Class MyCustomAPOLFX: public CBaseAudioProcessingObject

{

public:

//Code for custom class goes here

...

};

**Note**   因为SFX APO中的信号处理和MFX以及EFX APO存在较大差异, 应该为他们分别创建类.

1. 实现以下三个函数:
   * [**IAudioProcessingObject::IsInputFormatSupported**](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobject-isinputformatsupported). 此函数处理和音频引擎之间的格式协商.
   * [**IAudioProcessingObjectRT::APOProcess**](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess). 此函数使用自定义算法来执行信号处理.
   * **ValidateAndCacheConnectionInfo**. 此函数申请内存来存储格式细节, 比如通道数目, 采样率, 采样深度, 和通道掩码(channel count, sampling rate, sample depth, and channel mask).

以下C++示例代码展示了步骤1中提到的**[APOProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess)**函数的实现. 其具体的实现请参考上一节提到的Swap例子, 参考*Swapapolfx.cpp* 文件.

// Custom implementation of APOProcess method

STDMETHODIMP\_ (Void) MyCustomAPOLFX::APOProcess (...)

{

// Code for method goes here. This code is the algorithm that actually

// processes the digital audio signal.

...

}

以下C++示例代码展示了**ValidateAndCacheConnectionInfo** 函数的实现. 其具体的实现请参考上一节提到的Swap例子, 参考*Swapapogfx.cpp* 文件.

// Custom implementation of the ValidateAndCacheConnectionInfo method.

HRESULT CSwapAPOGFX::ValidateAndCacheConnectionInfo( ... )

{

// Code for method goes here.

// The code should validate the input/output format pair.

...

}

**注意**：Audioenginebaseapo.idl中详细描述了从CBaseAudioProcessingObject继承的那些接口和方法。

对于桌面PC, 可以提供一个UI来配置加到自定义APO中的那些功能. 更多信息, 请参考[实现配置APO的UI](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/implementing-a-ui-for-configuring-sapos).

## 2.5 替换系统提供的APO

实施APO接口时，有两种方式：您可以编写自己的实现，也可以调用系统内置(inbox))APO.

下面的伪代码展示了如果引用系统APO.

CMyWrapperAPO::CMyWrapperAPO {

CoCreateInstance(CLSID\_InboxAPO, m\_inbox);

}

CMyWrapperAPO::IsInputFormatSupported {

Return m\_inbox->IsInputFormatSupported(…);

}

下面的伪代码展示了如果实现自定义APO.

CMyFromScratchAPO::IsInputFormatSupported {

自定义逻辑

}

在开发APO来替换系统APO时，必须使用以下列表中的接口和函数名称。 除了列出的必需函数之外，有时还需要更多的函数。请参考相关链接, 以确定你需要实现哪些函数。其余的实现步骤与自定义APO相同。

为COM组件实现以下接口和函数:

* [IAudioProcessingObject](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudioprocessingobject). 这个接口里面需需要实现的函数是: [**Initialize**](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobject-initialize) 和 **[IsInputFormatSupported.](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobject-isinputformatsupported)**
* [IAudioProcessingObjectConfiguration](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudioprocessingobjectconfiguration). 这个接口里面实现的函数是: **[LockForProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectconfiguration-lockforprocess)** and **[UnlockForProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectconfiguration-unlockforprocess)**
* [IAudioProcessingObjectRT](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudioprocessingobjectrt). 这个接口里面需要实现函数[**APOProcess**](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectrt-apoprocess), DSP算法应该放在这里.
* [IAudioSystemEffects](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nn-audioenginebaseapo-iaudiosystemeffects). 该接口的目的是让音频引擎把我们的dll当做APO处理.

## 2.6 在Visual Studio中生成APO

使用Visual Studio来生成APO, 需要对每个工程执行以下任务.

### 2.6.1. 链接CRT

面向Win10的驱动应该动态链接通用C运行时库(universal CRT).

如果需要支持Windows 8.1, 在工程属性->C/C++->Code Generation中使能静态链接(static linking). 设置"Runtime Library"选项为*/MT*(release) 或 */MTd*(debug). 之所以要这么做是因为驱动包里面很难发布MSVCRT<n>.dll. 解决方案就是静态链接libcmt.dll. 更多信息请参考 [/MD, /MT, /LD (使用运行时库<Run-Time Library>)](https://docs.microsoft.com/cpp/build/reference/md-mt-ld-use-run-time-library) .

### 2.6.2. 去掉Embedded Manifest

通过设置APO工程的工程属性->**Manifest Tool**->**Input and Output**来去掉Embedded Manifest. 修改"Embed Manifest"从默认的Yes到No. 如果使用embedded manifest, 会触发使用一些特定的在保护模式(protected environment, 译注:[链接](https://docs.microsoft.com/en-us/windows/win32/medfound/protected-media-path#protected-environment))下被禁止的API. 那样的话APO在DisableProtectedAudioDG=1(译注: 注册表键值HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Audio\ DisableProtectedAudioDG)是工作正常; 但是一旦这个测试键值被去掉了, APO的载入就会失败, 即使WHQL sign也一样.

## 2.7 把APO打包到驱动

当您开发自己的音频驱动程序并包含或更换系统提供的APO时，必须提供用于安装驱动程序和APO的驱动程序包。 对于Windows 10，请参阅通用Windows音频驱动程序([Universal Windows Drivers for Audio](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/audio-universal-drivers))了解更多信息; 与音频相关的驱动程序包应遵循其中详细介绍的政策和打包模型。

自定义APO的形式是DLL，配置UI应该是独立的UWP或Desktop Bridge应用。 APO设备INF把DLL复制到相关的系统文件夹中(在INF的CopyFile指令中设定)。包含APO的dll会在INF文件中的AddReg部分进行注册。

以下部分显示了使用标准INF文件复制和注册APO所必需的修改。

Sysvad示例附带的tabletaudiosample.inf和phoneaudiosample.inf文件说明了如何注册SwapApo.dll APO。

## 2.8 在INF文件中注册APO的模式和音效位置

可以使用特定的注册表键值来注册APO, 并用于指定的模式. 关于有哪些音效是可用的, 以及更多的信息, 请参考APO架构章节.

关于APO INF文件的设置, 请参考以下相关的主题.

[PKEY\_FX\_StreamEffectClsid](https://docs.microsoft.com/windows-hardware/drivers/audio/pkey-fx-streameffectclsid)

[PKEY\_FX\_ModeEffectClsid](https://docs.microsoft.com/windows-hardware/drivers/audio/pkey-fx-modeeffectclsid)

[PKEY\_FX\_EndpointEffectClsid](https://docs.microsoft.com/windows-hardware/drivers/audio/pkey-fx-endpointeffectclsid)

[PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming](https://docs.microsoft.com/windows-hardware/drivers/audio/pkey-sfx-processingmodes-supported-for-streaming)

[PKEY\_MFX\_ProcessingModes\_Supported\_For\_Streaming](https://docs.microsoft.com/windows-hardware/drivers/audio/pkey-mfx-processingmodes-supported-for-streaming)

[PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming](https://docs.microsoft.com/windows-hardware/drivers/audio/pkey-efx-processingmodes-supported-for-streaming)

以下INF例子显示了如何为特定模式注册APO。 它们展示了本列表中可用的组合。

* PKEY\_FX\_StreamEffectClsid 和PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming
* PKEY\_FX\_ModeEffectClsid 和PKEY\_MFX\_ProcessingModes\_Suppoted\_For\_Streaming
* 只有PKEY\_FX\_ModeEffectClsid, 没有PKEY\_MFX\_ProcessingModes\_Suppoted\_For\_Streaming
* 只有PKEY\_FX\_EndpointEffectClsid, 没有PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming

这里示例中没有包含另外一种有效的组合.

* PKEY\_FX\_EndpointEffectClsid和 PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming

### 2.8.1. SYSVAD Tablet 多模式Mode 音效APO INF 例子

本例子展示了,在SYSVAD Tablet INF中,如何使用AddReg部分来注册多模式流音效(multi-mode streaming effect).

相关示例代码来自SYSVAD音频例子, 可以在github上找到: <https://github.com/Microsoft/Windows-driver-samples/tree/master/audio/sysvad>.

下面的例子展示了以下音效组合:

* PKEY\_FX\_StreamEffectClsid 和PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming
* PKEY\_FX\_ModeEffectClsid 和PKEY\_MFX\_ProcessingModes\_Suppoted\_For\_Streaming

[SWAPAPO.I.Association0.AddReg]

; 指示音频设备Builder(audio endpoint builder)把属性页 (译注:控制面板里面的UI) 的CLSID写入端点设备属性区(endpoint property store)

HKR,EP\0,%PKEY\_AudioEndpoint\_ControlPanelPageProvider%,,%AUDIOENDPOINT\_EXT\_UI\_CLSID%

; 指示音频设备Builder(audio endpoint builder)把APO的CLSID写入端点设备属性区(endpoint property store)

HKR,FX\0,%PKEY\_FX\_StreamEffectClsid%,,%FX\_STREAM\_CLSID%

HKR,FX\0,%PKEY\_FX\_ModeEffectClsid%,,%FX\_MODE\_CLSID%

HKR,FX\0,%PKEY\_FX\_UserInterfaceClsid%,,%FX\_UI\_CLSID%

; 驱动开发者需要修改下面的模式支持列表

; DEFAULT, MEDIA, MOVIE 模式的GUID集合

HKR,FX\0,%PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming%,%REG\_MULTI\_SZ%,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%,%AUDIO\_SIGNALPROCESSINGMODE\_MEDIA%,%AUDIO\_SIGNALPROCESSINGMODE\_MOVIE%

; DEFAULT, MEDIA, MOVIE 模式的GUID集合

HKR,FX\0,%PKEY\_MFX\_ProcessingModes\_Supported\_For\_Streaming%,%REG\_MULTI\_SZ%,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%,%AUDIO\_SIGNALPROCESSINGMODE\_MEDIA%,%AUDIO\_SIGNALPROCESSINGMODE\_MOVIE%

;HKR,FX\0,%PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming%,0x00010000,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%

请注意，在示例INF文件中，EFX\_Streaming属性被注释掉了(译注:只看到了EFX的mode设定被注释掉了, 没看到挂载EFX APO)，因为音频处理已转换到该层上方的内核模式(kernel mode above that layer, 译注:没理解, 是不是指算法都放到MFX上了?)，因此不需要流属性(streaming property)，并且将不使用流属性。 出于发现(discovery)的目的指定PKEY\_FX\_EndpointEffectClsid是有效的，但指定PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming是不对的。 这是因为模式mix/ tee在堆栈中的位置较低，因此无法插入端点APO (译注: 应该是指EFX本来就是放在对MFX后面的mixer之后的, EFX实际上处理的是mix了各模式的流, 没办法单独处理某些模式的流)。

### 2.8.2. Componentized APO 安装

从Windows 10版本1809开始，使用组件化的音频驱动程序来将APO注册到音频引擎。使音频组件化可提供更流畅，更可靠的安装体验，并更好地支持组件维护。更多信息，请参见创建组件化的音频驱动程序安装([Creating a componentized audio driver installation](https://docs.microsoft.com/windows-hardware/drivers/audio/audio-universal-drivers#creating-a-componentized-audio-driver-installation))。

以下代码来SYSVAD音频例子代码中的ComponentizedAudioSampleExtension.inf 和ComponentizedApoSample.inf. 相关代码可以在Github上找到: <https://github.com/Microsoft/Windows-driver-samples/tree/master/audio/sysvad>.

应使用新创建的APO设备来将APO注册到音频引擎. 音频引擎要求新的APO设备必须是音频设备的PNP(即插即用)子节点, 和音频端点并列.新的组件化APO设计不允许全局注册APO, 不允许在多个不同驱动中使用. 每个驱动必须注册自己的APO

APO的安装包括两部分. 首先驱动的extension INF会把APO设备添加到系统:

[DeviceExtension\_Install.Devices]

AddDevice = SwapApo,,Apo\_AddDevice

[Apo\_AddDevice]

HardwareIds = APO\VEN\_SMPL&CID\_APO

Description = "Audio Proxy APO Sample"

Capabilities = 0x00000008 ; SWDeviceCapabilitiesDriverRequired

这个APO设备会触发第二部分, 安装APO INF, 比如SYSVAD例子里面的ComponentizedApoSample.inf. 这个INF专用于APO设备. 它指定设备类为AudioProcessingObject，并添加所有APO属性以进行CLSID注册并向音频引擎注册。

[Version]

Signature = "$WINDOWS NT$"

Class = AudioProcessingObject

ClassGuid = {5989fce8-9cd0-467d-8a6a-5419e31529d4}

[ApoComponents.NT$ARCH$]

%Apo.ComponentDesc% = ApoComponent\_Install,APO\VEN\_SMPL&CID\_APO

[Apo\_AddReg]

; CLSID注册

HKCR,CLSID\%SWAP\_FX\_STREAM\_CLSID%,,,%SFX\_FriendlyName%

HKCR,CLSID\%SWAP\_FX\_STREAM\_CLSID%\InProcServer32,,0x00020000,%%SystemRoot%%\System32\swapapo.dll

HKCR,CLSID\%SWAP\_FX\_STREAM\_CLSID%\InProcServer32,ThreadingModel,,"Both"

…

;音频引擎注册

HKR,AudioEngine\AudioProcessingObjects\%SWAP\_FX\_STREAM\_CLSID%,"FriendlyName",,%SFX\_FriendlyName%

...

当该INF安装了组件化APO后, 在Windows设备管理器里面会看到system "Audio Processing Objects"(译注:按类型查看时, 在software components分类下面)

### 2.8.3. Bluetooth Audio APO INF 例子

这个例子描述了以下音效组合:

* PKEY\_FX\_StreamEffectClsid 和PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming
* PKEY\_FX\_ModeEffectClsid with PKEY\_MFX\_ProcessingModes\_Suppoted\_For\_Streaming

这个示例代码可以只蓝牙免提和立体声设备.

; wdma\_bt.inf – example usage

...

[BthA2DP]

Include=ks.inf, wdmaudio.inf, BtaMpm.inf

Needs=KS.Registration,WDMAUDIO.Registration,BtaMPM.CopyFilesOnly,mssysfx.CopyFilesAndRegister

...

[BTAudio.SysFx.Render]

HKR,"FX\\0",%PKEY\_ItemNameDisplay%,,%FX\_FriendlyName%

HKR,"FX\\0",%PKEY\_FX\_StreamEffectClsid%,,%FX\_STREAM\_CLSID%

HKR,"FX\\0",%PKEY\_FX\_ModeEffectClsid%,,%FX\_MODE\_CLSID%

HKR,"FX\\0",%PKEY\_FX\_UiClsid%,,%FX\_UI\_CLSID%

HKR,"FX\\0",%PKEY\_FX\_Association%,,%KSNODETYPE\_ANY%

HKR,"FX\\0",%PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming%,0x00010000,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%

HKR,"FX\\0",%PKEY\_MFX\_ProcessingModes\_Supported\_For\_Streaming%,0x00010000,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%

...

[Strings]

FX\_UI\_CLSID = "{5860E1C5-F95C-4a7a-8EC8-8AEF24F379A1}"

FX\_STREAM\_CLSID = "{62dc1a93-ae24-464c-a43e-452f824c4250}"

PKEY\_FX\_StreamEffectClsid = "{D04E05A6-594B-4fb6-A80D-01AF5EED7D1D},5"

PKEY\_FX\_ModeEffectClsid = "{D04E05A6-594B-4fb6-A80D-01AF5EED7D1D},6"

PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming = "{D3993A3F-99C2-4402-B5EC-A92A0367664B},5"

PKEY\_MFX\_ProcessingModes\_Supported\_For\_Streaming = "{D3993A3F-99C2-4402-B5EC-A92A0367664B},6"

AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT = "{C18E2F7E-933D-4965-B7D1-1EEF228D2AF3}"

### 2.8.4. APO INF例子

这个例子描述了以下音效组合:

* PKEY\_FX\_StreamEffectClsid 和PKEY\_SFX\_ProcessingModes\_Supported\_For\_Streaming
* PKEY\_FX\_ModeEffectClsid 和PKEY\_MFX\_ProcessingModes\_Suppoted\_For\_Streaming
* 只有PKEY\_FX\_EndpointEffectClsid , 没有PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming

[MyDevice.Interfaces]

AddInterface=%KSCATEGORY\_AUDIO%,%MyFilterName%,MyAudioInterface

[MyAudioInterface]

AddReg=MyAudioInterface.AddReg

[MyAudioInterface.AddReg]

;要注册APO让它能够被找到, 需要在inf文件中(或者在注册KSCATEGORY\_AUDIO 设备接口时<运行时>)使用以下属性键值 (原文:To register an APO for discovery, use the following property keys in the .inf (or at runtime when registering the KSCATEGORY\_AUDIO device interface):

HKR,"FX\\0",%PKEY\_FX\_StreamEffectClsid%,,%FX\_STREAM\_CLSID%

HKR,"FX\\0",%PKEY\_FX\_ModeEffectClsid%,,%FX\_MODE\_CLSID%

HKR,"FX\\0",%PKEY\_FX\_EndpointEffectClsid%,,%FX\_MODE\_CLSID%

;要注册APO让它能够被找到和分流, 也需要增加以下属性键值到同一个部分(原文:To register an APO for streaming and discovery, add the following property keys as well (to the same section)):

HKR,"FX\\0",%PKEY\_SFX\_ProcessingModes\_For\_Streaming%,%REG\_MULTI\_SZ%,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%,%AUDIO\_SIGNALPROCESSINGMODE\_MOVIE%,%AUDIO\_SIGNALPROCESSINGMODE\_COMMUNICATIONS%

;为了支持多模式处理, 需要创建一个多字符串键值, 并加上所有模式(原文:To register an APO for streaming in multiple modes, use a REG\_MULTI\_SZ property and include all the modes):

HKR,"FX\\0",%PKEY\_MFX\_ProcessingModes\_For\_Streaming%,%REG\_MULTI\_SZ%,%AUDIO\_SIGNALPROCESSINGMODE\_DEFAULT%,%AUDIO\_SIGNALPROCESSINGMODE\_MOVIE%,%AUDIO\_SIGNALPROCESSINGMODE\_COMMUNICATIONS%

### 2.8.5. 自定义APO 和CLSID APO INF 例子

这个例子展示了怎么给自定义APO定义CLSID. 其中使用了MsApoFxProxy CLSID {889C03C8-ABAD-4004-BF0A-BC7BB825E166}. CoCreate-ing这个GUID实例化了MsApoFxProxy.dll中的类, 该类实现了IAudioProcessingObject接口并通过KSPROPSETID\_AudioEffectsDiscovery属性集查询基础驱动程序。

This INF file sample shows the [BthHfAud] section, which pulls in [MsApoFxProxy.Registration] from wdmaudio.inf [BthHfAud.AnlgACapture.AddReg.Wave], which then registers PKEY\_FX\_EndpointEffectClsid as the well-known CLSID for MsApoFxProxy.dll.

此INF文件示例显示[BthHfAud]部分，该部分从wdmaudio.inf [BthHfAud.AnlgACapture.AddReg.Wave]中拉取(pull) [MsApoFxProxy.Registration]，然后将MsApoFxProxy.dll的CLSID注册到PKEY\_FX\_EndpointEffectClsid。

这个INF文件还展示了以下系统音效组合：

* 有PKEY\_FX\_EndpointEffectClsid , 无PKEY\_EFX\_ProcessingModes\_Supported\_For\_Streaming

;wdma\_bt.inf

[BthHfAud]

Include=ks.inf, wdmaudio.inf, BtaMpm.inf

Needs=KS.Registration, WDMAUDIO.Registration, BtaMPM.CopyFilesOnly, MsApoFxProxy.Registration

CopyFiles=BthHfAud.CopyList

AddReg=BthHfAud.AddReg

;在OEM inf中调用 (原文:Called by needs entry in oem inf)

[BthHfAudOEM.CopyFiles]

CopyFiles=BthHfAud.CopyList

[BthHfAud.AnlgACapture.AddReg.Wave]

HKR,,CLSID,,%KSProxy.CLSID%

HKR,"FX\\0",%PKEY\_FX\_Association%,,%KSNODETYPE\_ANY%

HKR,"FX\\0",%PKEY\_FX\_EndpointEffectClsid%,,%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%

下面的INF文件展示了 [MsApoFxProxy.Registration], 它调用了[MsApoFxProxy.AddReg]和[MsApoFxProxy.CopyList]. 前者注册了GUID和COM接口, 后者把MsApoFxProxy.dll拷贝到C:\Windows\system32.

; wdmaudio.inf –WmaLfxGfxDsp.dll在这里注册

...

;; MsApoFxProxy.Registration 区域会被OEM调用来安装APO

[MsApoFxProxy.Registration]

AddReg = MsApoFxProxy.AddReg

CopyFiles = MsApoFxProxy.CopyList

[MsApoFxProxy.CopyList]

MsApoFxProxy.dll,,,0x100

...

[MsApoFxProxy.AddReg]

; 音效APO COM接口注册 (原文:Discover Effects APO COM registration)

HKCR,CLSID\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%,,,%FX\_DiscoverEffectsApo\_FriendlyName%

HKCR,CLSID\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%\InProcServer32,,,%11%\MsApoFxProxy.dll

HKCR,CLSID\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%\InProcServer32,ThreadingModel,,"Both"

; 音效APO注册(原文:Discover Effects APO registration)

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "FriendlyName", ,%FX\_DiscoverEffectsApo\_FriendlyName%

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "Copyright", ,%MsCopyRight%

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MajorVersion", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MinorVersion", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "Flags", 0x00010001, 0x0000000d

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MinInputConnections", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MaxInputConnections", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MinOutputConnections", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MaxOutputConnections", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "MaxInstances", 0x00010001, 0xffffffff

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "NumAPOInterfaces", 0x00010001, 1

HKCR,AudioEngine\AudioProcessingObjects\%FX\_DISCOVER\_EFFECTS\_APO\_CLSID%, "APOInterface0", ,"{FD7F2B29-24D0-4B5C-B177-592C39F9CA10}"

...

## 2.9 APO的注册表

APO注册用于支持使用加权计算将效果与端点动态匹配的过程。加权计算使用以下属性仓库(property store)。每个音频接口有若干(>=0)个通过inf或运行时注册的端点属性仓库(*endpoint property stores* )和音效属性仓库(*effects property stores*).

最匹配的端点属性仓库(*endpoint property stores* )和音效属性仓库(*effects property stores*)有最高的权重, 会被使用. 其他的属性仓库会被忽略.

(译注: 没看懂这部分, 这里可能和注册表\HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4d36e96c-e325-11ce-bfc1-08002be10318}\<instance#>\ DeviceInterfaces\eSpeaker\DeviceParameters\下面的MSEP和MSFX键值有关, 这些键值下面会有名称为数字的)

匹配度按以下方式计算:

(译注: 这里还没搞懂, 感觉两个列表搞反了, 和类别名不一致)

端点属性仓库(*endpoint property stores* )权重

1. 适合特定设备类型(KSNODETYPE)的FX
2. 适合所有设备类型(KSNODETYPE\_ANY)的FX
3. 适合特定设备类型的MSFX
4. 适合所有设备类型(KSNODETYPE\_ANY)的MSFX

音效属性仓库(*effects property stores*)权重

1. 适合特定设备类型(KSNODETYPE)的EP
2. 适合所有设备类型(KSNODETYPE\_ANY)的EP
3. 适合特定设备类型的MSEP
4. 适合所有设备类型(KSNODETYPE\_ANY)的EP

属性仓库后面的数字必须从0开始, 并按顺序增加: MSEP\0, MSEP\1, …, MSEP\n

如果EP\3确实, Windows会停止查找后面的EP\n比如EP\4, 即使它们存在.

PKEY\_FX\_Association (端点属性仓库(*endpoint property stores* )) 或PKEY\_EP\_Association (音效属性仓库(*effects property stores*))被拿来与KSPINDESCRIPTOR.Category的值进行比较. 后者以内核流(Kernel Streaming)的形式表示信号链路末端的[pin factory](https://docs.microsoft.com/en-us/windows-hardware/drivers/audio/pin-factories)(filter可以实例化的所有pin). (The value of PKEY\_FX\_Association (for effects property stores) or PKEY\_EP\_Association (for endpoint property stores) is compared against the KSPINDESCRIPTOR.Category value for the pin factory at the hardware end of the signal path, as exposed by Kernel Streaming.)

只有微软内置的class driver(可以被第三方开发者引用)应该使用MSEP和MSFX, 第三方驱动应该使用EP和FX.

### 2.9.1. APO 节点类型兼容

以下示例文件说明了将PKEY\_FX\_Association键值设置为和APO关联的GUID.

;; 属性键值

PKEY\_FX\_Association = "{D04E05A6-594B-4fb6-A80D-01AF5EED7D1D},0"

"

;; 键值对

HKR,"FX\\0",%PKEY\_FX\_Association%,,%KSNODETYPE\_ANY%

因为音频适配器能够支持多个输入和输出，所以必须明确指出自定义APO兼容的内核流（KS）节点类型。 在前面的INF文件片段中，APO显示为与％KSNODETYPE\_ANY％的KS节点类型相关联。 在此INF文件的更高版本中，KSNODETYPE\_ANY的定义如下：

[Strings]

;; 定义在MyINF.inf中使用的字符串

...

KSNODETYPE\_ANY = "{00000000-0000-0000-0000-000000000000}"

KSNODETYPE\_SPEAKER = "{DFF21CE1-F70F-11D0-B917-00A0C9223196}"

...

KSNODETYPE\_ANY的NULL值表示此APO与任何类型的KS节点类型兼容。 例如，为了表明APO仅与KSNODETYPE\_SPEAKER的KS节点类型兼容，INF文件将显示KS节点类型和APO关联，如下所示：

;; 键值对

...

HKR,"FX\\0",%PKEY\_FX\_Association%,,%KSNODETYPE\_SPEAKER%

...

有关不同KS节点类型的GUID值的更多信息，请参见Ksmedia.h头文件。

## 2.10 APO载入失败分析

以下信息可帮助了解如何监视APO的故障。 可以使用这些信息对无法合并到音频图中的APO进行故障排除。

音频系统监视APO返回码，以确定APO是否已成功合并到图中。 它通过跟踪由任何一种指定方法返回的HRESULT值来监视返回代码。 对于每个已合并到图中的SFX，MFX和EFX APO，系统维护一个单独的故障计数值。

音频系统监视以下四种方法返回的HRESULT值。

* CoCreateInstance
* IsInputFormatSupported
* IsOutputFormatSupported
* LockForProcess

每当这些方法之一返回故障代码时，APO的故障计数值就会增加。 当APO返回表明已成功将其合并到音频链路中的代码时，故障计数将重置为零。 成功调用**[LockForProcess](https://docs.microsoft.com/windows/desktop/api/audioenginebaseapo/nf-audioenginebaseapo-iaudioprocessingobjectconfiguration-lockforprocess)**方法则表明APO已经导入成功。

[**CoCreateInstance**](https://docs.microsoft.com/windows/desktop/api/combaseapi/nf-combaseapi-cocreateinstance) 有点特别, 有很多原因会返回出错HRESUL. 三个最主要的原因如下:

* 链路在运行保护内容(protected content, 译注:[链接](https://docs.microsoft.com/en-us/windows/win32/medfound/protected-media-path#protected-environment)), 而APO没有正确签名.
* APO没注册.
* APO已改名或被篡改.

另外，如果SFX，MFX或EFX APO的失败计数值达到系统指定的限制， PKEY\_Endpoint\_Disable\_SysFx注册表项会被设置为“ 1”来禁用SFX，MFX和EFX APO。 系统指定的限制当前值为10。

# 3. 实现UI来配置APO音效

本文描述了如何实现一个UI界面一遍用户配置音效. 关于APO的更多信息, 请参考APO架构章节.

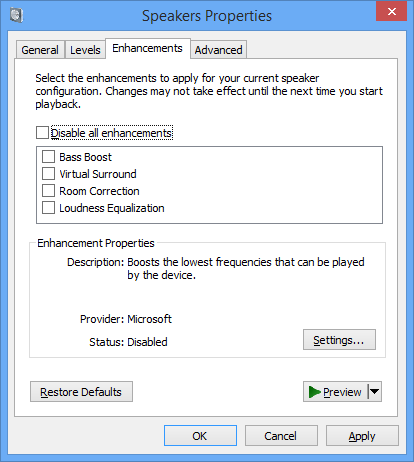
## 3.1. 概览

**注意**: 在Windows10 1809版之后，不再支持此自定义，并且PropPageExtensions项目不再出现在Sysvad示例中。对于更高版本的Windows，建议使用创建硬件支持应用程序的方法。有关更多信息，请参阅[硬件支持应用程序（HSA）：驱动程序开发人员的步骤](https://docs.microsoft.com/windows-hardware/drivers/devapps/hardware-support-app--hsa--steps-for-driver-developers)。

APO通常提供允许用户配置效果的UI。 例如，该UI可以允许用户从几种不同的信号处理算法中进行选择。 Microsoft提供了用于标准Windows APO的配置UI。 如果自定义APO具有用户可访问的设置，则开发人员必须提供适当的配置UI。 配置UI随设备驱动程序一起安装，并通过注册过程与APO关联。

注意制造商可以使用旨在支持其APO的自定义属性页替换此属性页。 如果他们的自定义APO没有用户可访问的设置，则制造商也可以选择完全不使用任何UI。

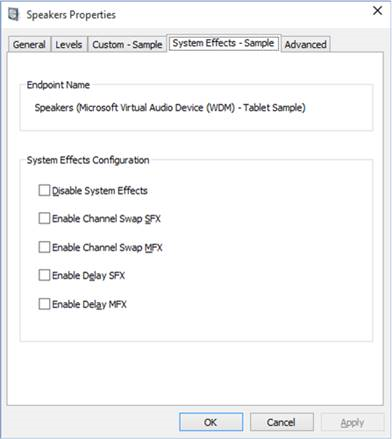
标准音效(enhancements,语音增强算法)选项卡如下所示。



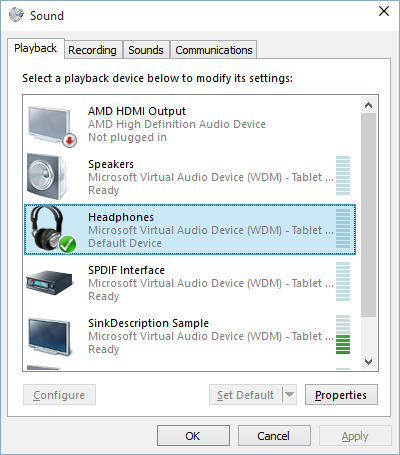
有三种可选方式来实现配置音效(语音增强算法)的UI：

* 完全不指定PKEY\_SYSFX\_UiClsid: 将不会显示任何音效属性页面(enhancements tab)。
* 将PKEY\_SYSFX\_UiClsid指定为标准系统自带的音效属性页面UI的CLSID: 将显示系统自带的音效属性页面。
* 使用自定义CLSID创建自定义的属性页，并将PKEY\_SYSFX\_UiClsid设置为自定义属性页的CLSID: 将显示自定义音效属性页面。

下图显示了SYSVAD Swap APO例子的的自定义属性页。



下图显示了控制面板中的声音小程序.



在“控制面板”的“声音”小程序中添加新的属性页，需要向系统提供的“声音”小程序添加新的选项卡。 这意味着在注册和初始化自定义APO时，自定义的属性页面和系统提供的音效(语音增强, enhancement)功能页面将并存。 在两个APO的属性页之间实现通信既困难又复杂。系统自带音效页面的某些默认设置可能会与新属性页面上的功能设置冲突。

因此，这里最实用的方法是实现一个单独的UI，以配置您开发的自定义APO来替换系统提供的APO。

## 3.2. 如何实现配置音效的UI

可以从音频端点的属性仓库(注册表)中获得音效UI的CLSID。 音频控制面板从当前音频端点属性仓库获取此CLSID。当音频控制面板启动适当的音效UI时，它将音频端点传递给它。 然后，UI可以访问端点属性仓库以读取和调整属性设置。 如果其他程序会修改设置，则UI还应该注册属性仓库通知。

如果使用CBaseAudioProcessingObject基类设计自定义APO并引用系统提供的APO，则可以替换默认属性页。

Windows在控制面板上为声音小程序提供了一个音效 (Enhancement) 属性页。 这是与系统内置音效APO相关联的默认属性页。 供应商可以实现和注册自定义属性页面，用自定义页面替换此默认属性页面。

关于如何替换音效 (Enhancement) 属性页面, 请参考属性页面([About Property Sheets](https://go.microsoft.com/fwlink/p/?linkid=106006)).

设计和实现定制属性页面提供程序的步骤:

1. 创建一个自定义属性页面。 更多信息，请参见创建属性表([Creating property sheets](https://go.microsoft.com/fwlink/p/?linkid=106006))。
2. 将自定义属性页打包为DLL。更多信息，请参考创建和使用DLL([Creating and using a DLL](https://go.microsoft.com/fwlink/p/?linkid=106014))。
3. 修改您的[INF文件](https://docs.microsoft.com/windows-hardware/drivers/install/overview-of-inf-files)以安装和注册属性页的DLL。

以下INF文件片段显示了如何修改INF文件以注册自定义属性页。

[SysFx.AddReg]

...

HKR,"FX\\0",%PKEY\_SYSFX\_UiClsid%,,%SYSFX\_UI\_CLSID%

...

[Strings]

PKEY\_SYSFX\_UiClsid = "{D04E05A6-594B-4FB6-A80D-01AF5EED7D1D},3"

SYSFX\_UI\_CLSID = "{自定义UI的GUID填在这里}"

以上INF文件中的指令将修改下列相应的注册表项.

HKLM

SOFTWARE

Microsoft

Windows

CurrentVersion

MMDevices

Audio

Render

Endpoint

FXProperties

...

"{D04E05A6-594B-4FB6-A80D-01Af5EED7D1D},3"=

"{自定义UI的GUID填在这里}"

默认属性页面的CLSID会被替换成自定义属性页面的CLSID

1. 在INF文件中全局AddReg部分, 注册CLSID相关的COM接口.

下面的INF例子来自SYSVAD tabletaudiosample.inf文件, 展示了相关细节. 这里的[SWAPAPO.AddReg]就是全局AddReg部分. [SWAPAPO.I.Association0.AddReg]是AddReg中用于特定KSCATEGORY\_AUDIO接口的部分.

[SWAPAPO.AddReg]

…

; 音效UI界面的COM接口注册

HKCR,CLSID\%FX\_UI\_CLSID%,,,"CplPage Class"

HKCR,CLSID\%FX\_UI\_CLSID%\InProcServer32,,,%11%\PropPageExt.dll

HKCR,CLSID\%FX\_UI\_CLSID%\InProcServer32,ThreadingModel,,"Apartment"

…

[SWAPAPO.I.Association0.AddReg]

…

HKR,FX\0,%PKEY\_FX\_UserInterfaceClsid%,,%FX\_UI\_CLSID%

…

[Strings]

FX\_UI\_CLSID = "{自定义UI的GUID填在这里}"

### 3.2.1. 使用或引用Windows的内置音效

如果在直接使用或者引用Windows提供的音效, 需要完成以下步骤:

请按照上面的步骤3，将自定义属性页的CLSID加到注册表中。

请按照上面的步骤4，注册COM接口。通过INF文件中的Include和Needs语句来调用wdmaudio.inf，如下所示。

[YourGlobalSection]

Include=wdmaudio.inf

Needs=mssysfx.CopyFilesAndRegister

## 3.3. SYSVAD SwapAPO UI Sample Code

使用SYVAD Swap APO例子代码作为模板可以加快自定义APO的开发过程。有关SWAP APO例子代码的更多信息，请参阅实现APO章节。

### 3.3.1. 示例代码

SYSVAD例子中的PropPageExtensions工程包含了APO属性页的例子代码.

|  |  |
| --- | --- |
| 工程 | 描述 |
| PropPageExtensions | 自定义属性页扩展UI的例子代码 |

了解以下文件对开发自定义UI会很有帮助:

|  |  |
| --- | --- |
| 名称 | 描述 |
| SwapPropPage.cpp | 实现CSwapPropPage类 |
| CplExt.cpp | 实现DLL 导出函数(Exports)和控制面板扩展 |
| UIWidgets.cpp | 实现CUIWidget和衍生类 |
| AdvEndpointPropPage.cpp | 实现CAdvEndpointPropPage |
| Parts.cpp | 实现CPart和衍生类 |
| TopologyExaminers.cpp | 实现支持检查音频拓扑的方法，例如连接器和端点。 |

为了熟悉PropPageExtensions例子代码，可能需要多查看相关头文件，然后检查和在属性页上定义文本相关的源代码。 如果需求与示例代码提供的功能相似，则可以重用许多用于创建和更新自定义UI页面的代码。

# 4. 实现硬件分担(Hardware Offloaded) APO音效

相关内容在较早文件中翻译了. 请参考: [实现硬件分担(Hardware Offloaded) APO音效](https://blog.csdn.net/danteLiujie/article/details/100707708)

# 5. 微软默认APO (缺)

将会另起一文翻译