

## DirectShow SDK Programmer's Guide

## Contents

DirectShow SDK Programmer's Guide .....	1
Contents.....	2
Overview .....	3
Vision Kernel Video streaming source filters .....	3
Vision User-mode Video streaming source filters .....	3
Vision User-mode Audio streaming source filters .....	3
Kernel video streaming vs. User-mode video streaming .....	4
SDK Contents.....	5
Interface Reference.....	10
Kernel Video source filters .....	11
User-mode Video source filters .....	11
User-mode Audio source filters .....	11

## Overview

DirectShow is a Microsoft Windows technology used as an interactive component approach to data transfer amongst third party device vendors. Microsoft DirectShow is regarded as the best approach to saving encoded data to disk using the Windows operating system. The DirectShow interface can be broken into 'Pin' and 'Device' centric operations. DirectShow uses 'Filters' connected within a 'Filter Graph' that transfer data from source to destination. Typically, three types of filter exist:

Source

Transform

Render

An example filter graph may contain ' Vision Capture source filter -> Subtitle Transform Filter -> Video Renderer'. Many combinations of filter exist in DirectShow graphs including audio capture, transform encoders, transform decoders and file writers. The Datapath Vision capture filter supports the generic Microsoft IAMStreamConfig interface. The Vision source filter also exposes custom 'filter' and 'pin' property pages in addition to hardware specific IVision and IVisionEvent interfaces.

A Vision driver exposes itself as a CLSID\_VideoInputDeviceCategory source 'filter' containing two output 'pins' Capture and Preview.

The Vision installation allows the selection of either Kernel-mode or User-mode source filters.

### **Vision Kernel Video streaming source filters**

Devices are represented as a WDM capture source 'filters' containing two output 'pins', Capture and Preview and a single input 'pin' for Crossbar connection. A total of 16 instances of the source filter can be instantiated to provide 16 independent streams across multiple processes if required. The capture filter supports the generic Microsoft IAMStreamConfig interface. The source filter also exposes custom 'filter' and 'pin' property pages in addition to hardware specific IVision and IVisionEvent interfaces. The input pin supports the IAMCrossbar interface.

### **Vision User-mode Video streaming source filters**

Inputs are represented as a video capture source 'filters' containing two output 'pins', Capture and Preview. A total of 8 instances of the source filter can be instantiated to provide 16 independent streams across multiple processes if required. The capture filter supports the generic Microsoft IAMStreamConfig interface. The source filter also exposes custom 'filter' and 'pin' property pages in addition to hardware specific IVisionUser and IVisionUserEvent interfaces.

### **Vision User-mode Audio streaming source filters**

Inputs are represented as a video capture source 'filters' containing one output 'pin'. The capture filter supports the generic Microsoft IAMStreamConfig interface. The source filter also exposes custom 'filter' property pages in addition to a hardware specific IAudioUser interface.

## **Kernel video streaming vs. User-mode video streaming**

A system containing a VisionSD4+1 and a VisionRGB-E2S will appear as the following source filters using Kernel Video streaming:

VisionSD4+1 01

VisionSD4+1 02

VisionRGB-E2S 03

VisionRGB-E2S 04

Where the 'VisionSD4+1 02' filter represents four SD inputs that will require a Crossbar for access.

The same system will appear as the following source filters using User-mode Video streaming:

VisionSD4+1 Video 01

VisionSD4+1 Video 02

VisionSD4+1 Video 03

VisionSD4+1 Video 04

VisionSD4+1 Video 05

VisionRGB-E2S Video 06

VisionRGB-E2S Video 07

Notice how the four SD inputs are represented by individual source filters. Crossbars are not required.

## SDK Contents

### Example Compiler Settings

Set your compilers library settings to `/libpath:"D:\DXSDK\LIB"`, the local DirectX SDK lib directory.

Set your compilers include settings to `/I "D:\DXSDK\INCLUDE"`, the local DirectX SDK include directory.

To compile the SDK applications for use with User-mode filters, include `USER_MODE_STREAMING` in the project's preprocessor definitions for all configurations. Project files in the form of `SampleX` are configured to build the samples using the kernel source filters. Project files in the form of `SampleX_um` are configured to build the samples using the User-mode source filters.

### COMMON

Common source files.

### INCLUDE

Common header files.

### Sample1

Simple capture graph composition.

### Sample2

Programmatically set properties of the Filter.

Programmatically set properties of the Pin.

Load the `IVision` interface.

Load the `IVisionEvent2` interface.

Set output buffer size and client area of window to input resolution dynamically on mode changes.

### Sample3

Display the capture card plug-in Filter Properties dialogue.

Display the capture card plug-in alias Pin Properties dialogue.

### Sample4

Parse and programmatically set the command line.

Example: `-input=1 -fps=60 -subformat=RGB32 -buffer=640,480 -window=30,20,640,480`

BatchFiles

Use with `Sample4.exe`. See `.bat` files for further information.

### Sample5

Configure and save .avi files to disk.

Configure and save .wmv files to disk.

Load and use the IAMStreamControl interface with User-mode filters.

See TODO: for configuration of your preferred audio capture device.

### Sample6

Uses VIDEOINFOHEADER2 to describe interlaced video sources.

Load and use VMR9 to de-interlace the media format.

Load and use IAMCrossbar interface for kernel filters.

Load and use the IVisionUserPin interface for cropping with User-mode filters.

### SampleAV

Audio and Video Synchronisation Sample

## NullRenderer

Attach this renderer filter to the capture card source filter and calculate the system capable frame rate prior to additional downstream filter processing.

The DGC Null Renderer can be used to calculate the data rate transferred across the system bus out of the capture card into system memory. The frame rates achieved reflect the system hardware capabilities ignoring software buffer manipulation and rendering techniques.

Attach your source to input 1 on your capture card.

Open graphEdt.exe found within your DXSDK\Bin\DXUtils folder.

Insert a Vision capture source filter.

Open the "Filter Properties", right click the filter and select properties.

Press "Reset" in the dialogue and ensure the width and height reflect the source resolution.

Close the "Filter Properties" dialogue.

Open the "Pin Properties", right click the pin at the side of the filter.

Ensure that the default buffer is the source resolution for a 1:1 capture/buffer.

Select your preferred frame rate, the default is set to the current input signal frequency.

Select one of the following colour space values:

RGB8 - 1 Byte per pixel

RGB565 - 2 Byte per pixel

RGB24 - 3 Byte per pixel

RGB32 - 4 Byte per pixel

Set the output buffer size, the 'Default Size' is the currently connected resolution, use this for 1:1 no scaling.

Close the "Pin Properties" dialogue.

Insert the "DGC Null Video Renderer" filter, "Graph->Insert->Filters->DirectShow Filters-> "DGC Null Video Renderer".

Hold the left mouse button down over the Vision capture source filter pin and manually connect the arrow to the "DGC Null Video Renderer".

Select "Graph->Play".

Right click the "DGC Null Video Renderer" filter and select "Filter Properties" from the context menu.

The "Average frame rate achieved" field represents the frames delivered to the null renderer from the capture source filter.

## DiagTransform

Use the 'DGC Diagnostics Filter' to inspect the time stamp and associated clocks of a 'Capture' pin stream.

Reference Time - The system clock used as a base line (zero 'd) to the stream clock under default conditions.

Stream Clock - The base line clock for the graph and streams, starts from zero.

Stamp Time - The actual frame start time stamp.

Difference - The difference between the Stream Clock and Stamp Time.

Stop - The frame end time.

Start/Stop Difference - The difference between the Stamp Time and Stop.

Successive Frame Interval - Difference on the input pin between successive frame time stamps.

All units are in 100ns 'ticks'.

By programmatically setting SetSyncSource(NULL) or using GraphEdt.exe 'Graph->Use Clocks == OFF' the Stream Clock will remain as zero and the time stamps will be relative to the KeQueryPerformanceCounter timer as specified in the 'Capture Time Stamps' paragraph above.

Attach your source to input 1 on your capture card.

Open graphEdit found within your DXSDK\Bin\DXUtils folder.

Insert a capture source filter.

Open the "Pin Properties", right click the pin at the side of the filter.

Select your preferred frame rate, the default is set to the current input signal frequency.

Select your preferred colour space compression.

Close the "Pin Properties" dialogue.

Insert the "DGC diagnostics transform" filter, "Graph->Insert->Filters->DirectShow Filters->DGC diagnostics transform.

Hold the left mouse button down over the DGC diagnostics transform filter output pin and connect to your preferred Renderer.

Select "Graph->Play".

Right click the "DGC diagnostics transform" filter and select "Filter Properties" from the context menu.

A running result for 'Capture Pin' time stamps diagnostics is displayed within the dialogue.

For a complete listing of diagnostics from graph start time, see the file log created at the path specified on the dialogue.

"Graph time" CBaseFilter::StreamTime(rtStream);

"start" GetTime((REFERENCE\_TIME \*) &tStart, (REFERENCE\_TIME \*) &tStop)

"clock diff" start-Graph time

"stop" GetTime((REFERENCE\_TIME \*) &tStart, (REFERENCE\_TIME \*) &tStop)

"start/stop diff" stop-start

"pin delivery" start-start-1

## Interface Reference

The 'Pin' properties of the Vision source filter exposes 14 data formats for selection.

The available formats are 7 compressions, YUY2, YVYU, UYVY, RGB8, RGB565, RGB24, and RGB32 in both VIDEOINFOHEADER and VIDEOINFOHEADER2 formats. VIDEOINFOHEADER2 is used for de-interlacing interlaced video sources.

Sample6 of the Vision DirectShow SDK shows how to de-interlace interlaced video sources using your display device.

The default bitmap header defining the available data formats represent the input signal resolution. For the case of 'No Signal', 320x240 is used.

The output buffer size can be set at any time using the bitmap info structure contained within the data formats, see SetBufferSize() within VisionPin.c as part of the SDK.

The IAMStreamConfig->VIDEO\_STREAM\_CONFIG\_CAPS parameters MinOutputSize and MaxOutputSize reflect the minimum and maximum output buffer frame size.

To set the buffer size use the MEDIATYPE->SampleSize and MEDIATYPE->VIDEOINFOHEADER->BitmapInfoHeader data fields, then call IAMStreamConfig->SetFormat()

The OutputGranularityX and OutputGranularityY values represent buffer alignment, for example, if OutputGranularityX == 8, all buffer widths must be divisible by 8 with no remainder. For further details see the 'Format Granularity' section below.

For frame rates, use any range between the MinFrameInterval and MaxFrameInterval from the VIDEO\_STREAM\_CONFIG\_CAPS structure. To set a frame rate use the MEDIATYPE->VIDEOINFOHEADER->AvgTimePerFrame data field and call IAMStreamConfig->SetFormat()

Please see the 'Remarks' section for Microsoft VIDEO\_STREAM\_CONFIG\_CAPS Structure documentation for further details: [http://msdn.microsoft.com/en-us/library/dd407352\(v=vs.85\).aspx](http://msdn.microsoft.com/en-us/library/dd407352(v=vs.85).aspx)

The Directshow SDK for Vision capture cards consists of six sample applications that demonstrate the use of the Kernel and user-mode filters and their interfaces.

Both the Kernel and User-mode source filters expose interfaces to assist software developers:

## Kernel Video source filters

[IAMStreamConfig](#) - Generic Microsoft interface. Use to access the video format.

[IAMCrossbar](#) - Generic Microsoft interface. Use to select individual inputs from a device.

IVision - Access to device specific parameters.

IDGC103ControlConfig - for backward compatibility with Vision RGB devices.

IVisionEvent - Obsolete, use IVisionEvent2.

IVisionEvent2 - Call back functionality for 'No Signal', 'Mode Change' and 'Parameters Changed'.

IVisionCallBack - Call back class used for IVisionEvent2.

## User-mode Video source filters

[IAMStreamConfig](#) - Generic Microsoft interface. Use to access the video format.

[IAMStreamControl](#)- Generic Microsoft interface. Use to control individual streams.

IVisionUser - Access to input specific parameters.

IVisionUserEvent - Call back functionality for 'No Signal', 'Mode Change' and 'Parameters Changed'.

IVisionCallBack - Call back class used for IVisionUserEvent.

IVisionUserPin - Use to access pin specific parameters.

## User-mode Audio source filters

[IAMStreamConfig](#) - Generic Microsoft interface. Use to access the video format.

[IAMStreamControl](#)- Generic Microsoft interface. Use to control individual streams.

IAudioUser - Access to input specific parameters.

Reference for the Vision specific interfaces can be found in the type-library (\*.tlb) files located in  
<SDKDIR>\DIRECTSHOW\DOCS