June 2022
Developer Information

Blackmagicdesign

Blackmagic
RAW SDK

Blackmagic RAW SDK

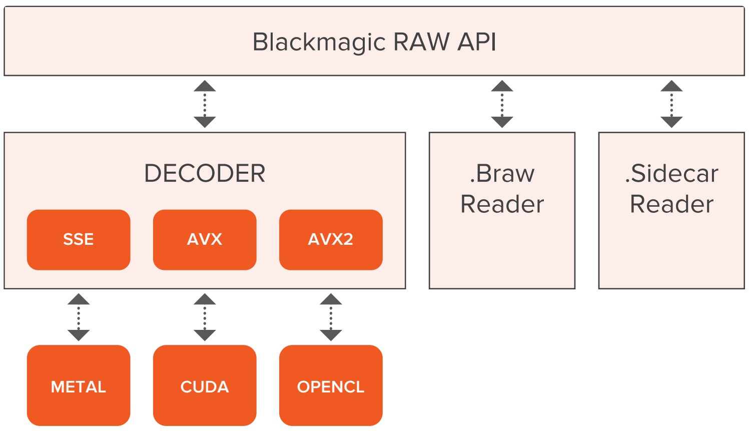
Contents

Blackmagic RAW SDK

Introduction and Overview

1.0 API Overview

The Blackmagic RAW SDK provides a highly optimised decoder and image processing pipeline.



Available on Mac, Windows, and Linux platforms, the SDK supports supports multiple CPU architectures and multiple GPU APIs in order to take full advantage of your machine.

The goal is to provide an easy to use yet powerful SDK, which will utilise cross-platform efficient decoding of .braw files produced by Blackmagic Cameras.

1.1 Decoder Overview

The CPU decoder has been designed to scale from laptops to workstations with a large number of cores. The CPU decoder will utilise SSE, AVX and AVX2 instructions if available. The user has control to limit the CPU decoder to fewer threads if desired.

There are several GPU decoders available, including Metal, CUDA, and OpenCL. The final processed image from each decoder will be provided in a buffer object native to the respective GPU API. This will allow quick access for further processing or display.

The GPU decoders are dynamically loaded, meaning they will require the system to have the relevant APIs installed in order to function.

The SDK has been designed for multi-GPU and multi-process capabilities allowing high level workstations to use all the resources available in the system.

1.2 Sidecar

A .sidecar file may be used, storing any metadata that is modified after the original .braw file is produced. The intent here is to not modify the original .braw file. This sidecar file can be manually deleted if the user wants to restore the movie metadata to its original state.

When metadata or image processing values (such as white balance) are modified via the SDK, the user can then choose to save this data to the sidecar file. Now when the movie is loaded (potentially in a different application) the sidecar file be applied and the picture will look consistent.

At this time, the user can run the 'trim' operation which will bake the sidecar changes into a newly created .braw file saved to disk. This can be run on any frame range right down to a single frame which produces handy images to pass between colleagues.

The .sidecar file is stored as a text JSON file, allowing users to manually edit or use external tools if they wish to modify it.

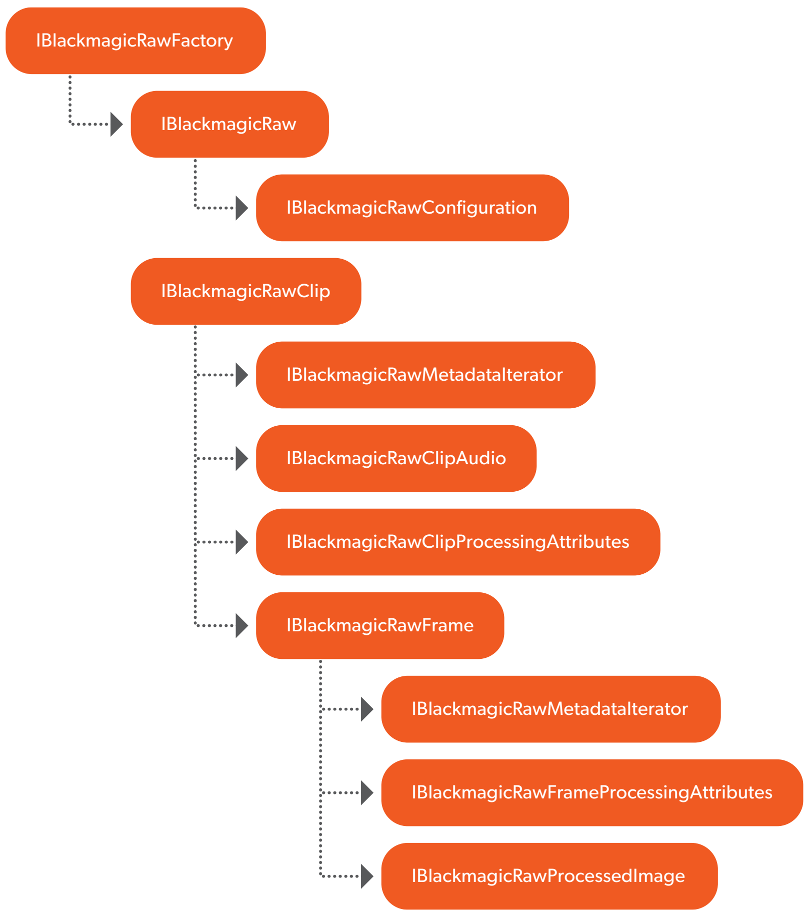
3DLUT data (in DaVinci Resolve .cube format) can be embedded into .braw clips and also be stored in sidecar files. This provides additional ways for 3DLUTs to travel with .braw clips to maintain consistency in viewing. The SDK allows users to optionally process the clip with the embedded 3DLUT in the clip itself, from the sidecar, or for 3DLUT processing to be disabled. Tetrahedral interpolation is used for both GPU and CPU pipelines.

An example sidecar with an identity 3DLUT follows. The information has been truncated for brevity.

```
{

    "tone_curve_contrast"              : 1.450000,
    "tone_curve_saturation"            : 1.150000,
    "tone_curve_midpoint"              : 0.409000,
    "tone_curve_highlights"            : 0.600000,
    "tone_curve_shadows"               : 1.800000,
    "tone_curve_black_level"           : 0.000000,
    "tone_curve_white_level"           : 1.000000,
    "tone_curve_video_black_level"     : 1,
    "highlight_recovery"               : 1,
    "viewing_gamma"                    : "Blackmagic Design Custom",
    "viewing_gamut"                    : "Blackmagic Design",
    "exposure": {
        "14:57:33:00"                  : 0.200000
    },
    "white_balance_kelvin": {
        "14:57:33:00"                  : 4520
    },
    "white_balance_tint": {
        "14:57:33:00"                  : 5
    },


    "iso": {
        "14:57:33:00"                  : 500
    },
    "post_3dlut_mode"                  : "Sidecar",
    "post_3dlut_sidecar_name"          : "Identity3DLUT.cube",
    "post_3dlut_sidecar_title"         : "My Identity 3D LUT",
    "post_3dlut_sidecar_size"          : 33,
    "post_3dlut_sidecar_data"          : "0.0000000000 0.0000000000 0.0000000000
                                         0.0312500000 0.0000000000 0.0000000000
                                         0.0625000000 0.0000000000 0.0000000000
                                         ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
                                         0.9375000000 1.0000000000 1.0000000000
                                         0.9687500000 1.0000000000 1.0000000000
                                         1.0000000000 1.0000000000 1.0000000000"

}
```

2.0 Interface Overview

This chapter covers a basic overview of the interfaces used via the Blackmagic RAW API


IBlackmagicRawFactory
IBlackmagicRaw
IBlackmagicRawConfiguration
IBlackmagicRawClip
IBlackmagicRawMetadataIterator
IBlackmagicRawClipAudio
IBlackmagicRawClipProcessingAttributes
IBlackmagicRawFrame
IBlackmagicRawMetadataIterator
IBlackmagicRawFrameProcessingAttributes
IBlackmagicRawProcessedImage

IBlackmagicRawFactory is the API entry point. From here the user creates a *IBlackmagicRaw* object. This object owns a single decoder instance.

This object can be configured via *IBlackmagicRawConfiguration* allowing the user to define CPU constraints or set up the SDK to use desired GPU APIs.

After completing the above steps, the user can start opening clips. Once the first clip has been opened, the decoder is started and any further configuration changes will be discarded.

2.2 Clip Object

Once a clip is opened its components can now be accessed. A metadata iterator is available to provide all clip-level metadata (see frame-level metadata in 2.3 below).

Clip audio & clip-level processing attributes can also be accessed. The clip-level processing attributes allow the user to modify fields such as displayed gamma, displayed gamut, Blackmagic colour science generation and custom gamma parameters.

Finally with the clip object we create an asynchronous job to read a frame from the clip. This will provide a frame object.

2.3 Frame Object

A frame object provides access to frame-level metadata & frame-level processing attributes. The frame-level processing attributes allow the user to modify fields that can change on a per-frame basis. They include white balance tint/kelvin, exposure & ISO.

The user can also specify output scale & the desired pixel format of the processed image which is produced upon request from the frame.

Once ready, the user creates an asynchronous job to produce the processed image. This image is then ready for display.

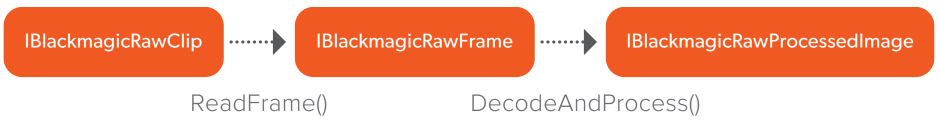
3.0 SDK Operations and flow

This chapter provides a brief explanation of how the above objects work together to produce a final image.

The SDK provides three main operations, read, decode and process. The read operation is reading the compressed image from an opened *IBlackmagicRawClip* file into a *IBlackmagicRawFrame*. The decode operation decodes this compressed image format and prepares it for processing. The process operation applies colour processing (such as white balancing, exposure) and provides the final image.

Each of these operations are asynchronous and occur across multiple CPU threads / GPU contexts. By default this is all handled internally to provide an easy yet efficient solution.

The flow described above is: open a clip, read a frame, decode and process frame:



3.1 Manual Decoders

There are manual decoders available which split the 3 above operations into separated user-driven steps. These are for advanced use and provide closer access to buffer control, memory use, GPU contexts and so forth, should your application require it.

Please see the *IBlackmagicRawManualDecoder** interfaces available in the API header to use this approach.

4.0 GPU Configuration

When utilising the GPU, at configuration time (described above in section 2.0) the GPU devices must be provided to the *IBlackmagicRawConfiguration* object. This includes passing in a context and commandQueue for the desired device.

To create a context and *commandQueue* the user needs to utilise their desired compute API library (i.e. Metal, CUDA or OpenCL).

To make this easier for the user, Blackmagic RAW SDK offers pipeline and device iterators. These allow creation of devices in an abstract way, removing the need to deal directly with compute APIs.

4.1 Pipeline iterators

Iterating through the available pipelines will allow your application to query for the presence and usability of CUDA, Metal, OpenCL and CPU based decoder pipelines. Each of these may be used to create a pipeline device iterator, with which associated compatible devices may be created.

Using this interface allows applications to check for various compute APIs without the need to set-up and call any of the API functions and hence removes the requirement of linking against API libraries and associated dependencies explicitly.

The pipeline iterator is created via *IBlackmagicRawFactory's* CreatePipelineIterator method.

4.2 Pipeline device iterators

The pipeline device iterator (IBlackmagicRawPipelineDeviceIterator) is used to iterate over all devices that a specific pipeline supports.

Once a (*IBlackmagicRawPipelineDevice*) device has been created via the device iterator, it may be used to configure a decoder with the SetFromDevice method of *IBlackmagicRawConfiguration*.

This is equivalent to, but more convenient than, querying the context and command queue from the device with the GetPipeline method and providing these parameters to the decoder configuration via SetPipeline.

*The context and command queue owned by a device are compute-level API objects that are used directly by the decoder. The life-time of these compute-level API objects must outlive that of the decoder, therefore the device instance **MUST** outlive the decoder instance.*

4.3 Pipeline preparation

A pipeline may have resources which require preparation, such as compilation or binding of GPU kernels to a device. The SDK provides a mechanism whereby the user may prepare these resources ahead of time, removing any stall necessitated by the use of these resources in the actual decoding process.

The preparation of these resources is a potentially time-consuming process and as such is executed asynchronously. The callback interface has a method (*PreparePipelineComplete*) to allow the user to respond to the completion of a pipeline preparation.

A pipeline is prepared on *IBlackmagicRaw* with either PreparePipeline (providing context and command queue) or *PreparePipelineForDevice* (providing an instance of *IBlackmagicRawPipelineDevice*), noting that the pipeline configuration MUST have been set prior.

4.4 Multi GPU Devices

When using multiple GPUs, a default GPU device is provided to the *IBlackmagicRawConfiguration* object.

Taking advantage of the manual decoders (specified in section 3.1 above) allows the user to distribute decoding operations across multiple devices.

Recommended UI Controls and Behavior

Decode Quality

Type:	**Drop down selector**
Default:	Highest value.
Options:	Use IBlackmagicRawClipResolutions::GetResolution()

Color Science Version

Type:	**Drop down selector**
Default:	Read from metadata
Options:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeList() – blackmagicRawClipProcessingAttributeColorScienceGen

Color Space/Gamut

Type:	**Drop down selector**
Default:	Read from metadata
Options:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeList() – blackmagicRawClipProcessingAttributeGamut

Gamma

Type:	**Drop down selector**
Default:	Read from metadata.
Options:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeList() – blackmagicRawClipProcessingAttributeGamma

ISO

Type:	**Drop down selector**
Default:	Read from metadata.
Options:	Use IBlackmagicRawClipProcessingAttributes::GetISOList()

Exposure

Type:	**Slider**
Default:	0.
Range:	Use IBlackmagicRawFrameProcessingAttributes::GetFrameAttributeRange() – blackmagicRawFrameProcessingAttributeExposure

Color Temp

Type:	**Slider**
Default:	5600
Range:	Use IBlackmagicRawFrameProcessingAttributes::GetFrameAttributeRange() – blackmagicRawFrameProcessingAttributeWhiteBalanceKelvin

Tint

Type:	**Slider**
Default:	10
Range:	Use IBlackmagicRawFrameProcessingAttributes::GetFrameAttributeRange() – blackmagicRawFrameProcessingAttributeWhiteBalanceTint

Highlight Recovery

Type:	**Checkbox**
Default:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeHighlightRecovery

Gamut Compression

Type:	**Checkbox**
Default:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeGamutCompressionEnable

Export Frame

Type:	**Button**

Exports a single frame of the currently viewed video frame.

Update Sidecar

Type:	**Button**

Saves sidecar file with the currently set parameters for the clip.

Custom Gamma Controls

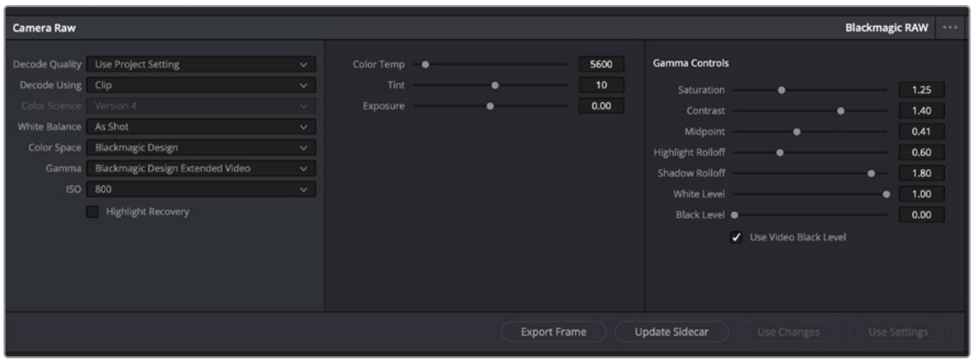
Custom gamma controls should only be enabled and selectable for the following gamma selections:

- **Blackmagic Design Film**
- **Blackmagic Design Extended Video**
- **Blackmagic Design Custom**

NOTE: **Blackmagic Design Video** should have the custom gamma controls DISABLED.

When selecting **Blackmagic Design Film** or **Blackmagic Design Extended Video** the custom gamma controls take on the values supplied by the SDK. When a user adjusts a custom gamma control slider, the gamma selection should automatically change to **Blackmagic Design Custom** which should be written with the current values shown in the UI. The user is now creating their own custom gamma which can be stored.

The following image examples show the custom gamma controls selectable with Blackmagic Design Film, and disabled with an incompatible gamma such as Rec.709.

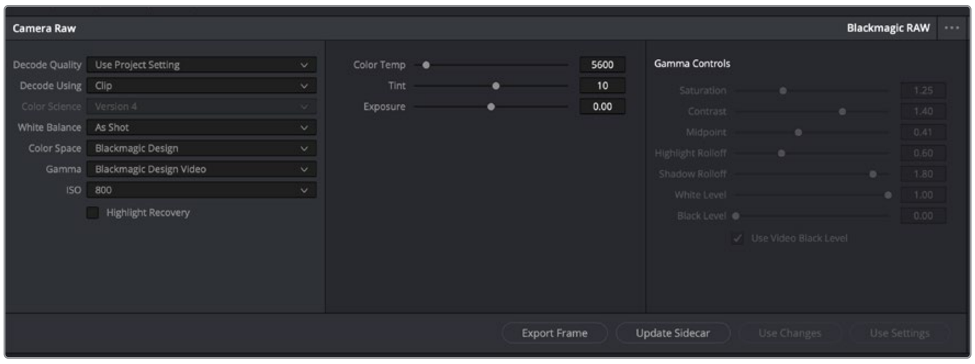


Decode Quality: Use Project Setting
Decode Using: Clip
Color Science: Version 4
White Balance: As Shot
Color Space: Blackmagic Design
Gamma: Blackmagic Design Extended Video
ISO: 800
Highlight Recovery

Color Temp: 5600
Tint: 10
Exposure: 0.00

Saturation: 1.25
Contrast: 1.40
Midpoint: 0.41
Highlight Rolloff: 0.60
Shadow Rolloff: 1.80
White Level: 1.00
Black Level: 0.00
✓ Use Video Black Level

Example: The custom gamma controls (last 3rd of the RAW panel) are enabled and selectable with Blackmagic Design Extended Video.



Decode Quality: Use Project Setting
Decode Using: Clip
Color Science: Version 4
White Balance: As Shot
Color Space: Blackmagic Design
Gamma: Blackmagic Design Video
ISO: 800
Highlight Recovery

Color Temp: 5600
Tint: 10
Exposure: 0.00

Saturation: 1.25
Contrast: 1.40
Midpoint: 0.41
Highlight Rolloff: 0.60
Shadow Rolloff: 1.80
White Level: 1.00
Black Level: 0.00
✓ Use Video Black Level

Example: The custom gamma controls (last 3rd of the RAW panel) are disabled with Blackmagic Design Video gamma.

Saturation

Type: **Slider**

Default: 1.0

Range: Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange()
– blackmagicRawClipProcessingAttributeToneCurveSaturation

Contrast

Type: **Slider**

Default: 0.5

Range: Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange()
– blackmagicRawClipProcessingAttributeToneCurveContrast

Midpoint

Type: **Slider**

Default: 0.41

Range: Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange()
– blackmagicRawClipProcessingAttributeToneCurveMidpoint

Highlight Rolloff

Type:	**Slider**
Default:	1.0
Range:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeToneCurveHighlights

Shadow Rolloff

Type:	**Slider**
Default:	1.0
Range:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeToneCurveShadows

Black Level

Type:	**Slider**
Default:	1.0
Range:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeToneCurveBlackLevel

White Level

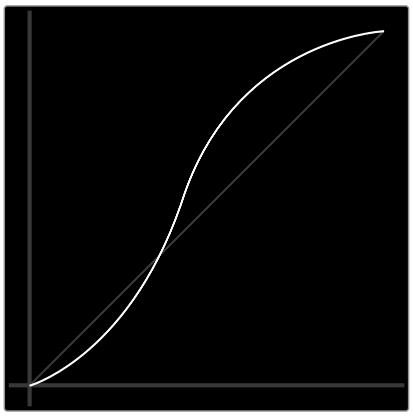
Type:	**Slider**
Default:	1.0
Range:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeToneCurveWhiteLevel

Set Video Black Level

Type:	**Checkbox**
Default:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange() – blackmagicRawClipProcessingAttributeToneCurveVideoBlackLevel

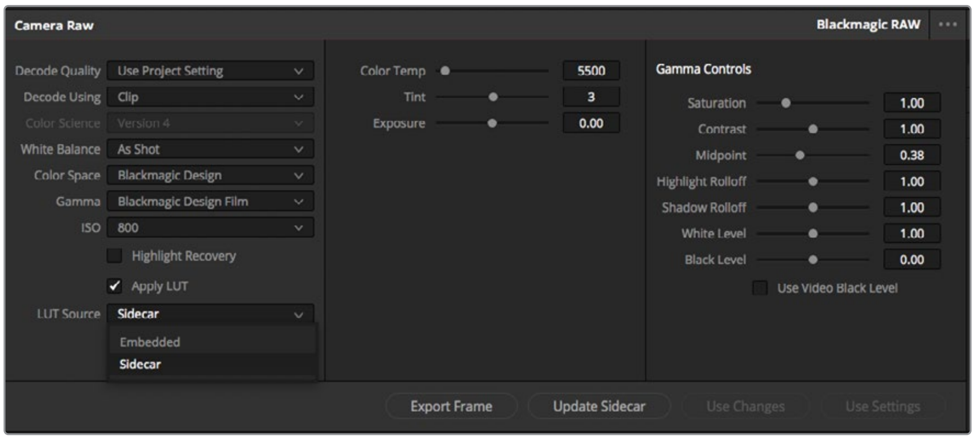
IBlackmagicRawToneCurve::EvaluateToneCurve()

The EvaluateToneCurve method can be used to return a buffer that can then be used to draw and visualize the result of the custom gamma controls. This is particularly useful when users are creating their own custom gammas. An example of such a UI is shown below:



3D LUT

Type:	**Drop down selector**
Default:	Read from metadata.
Options:	Use IBlackmagicRawClipProcessingAttributes::GetClipAttributeList() – blackmagicRawClipProcessingAttributePost3DLUTMode



Decode Quality: Use Project Setting
Decode Using: Clip
Color Science: Version 4
White Balance: As Shot
Color Space: Blackmagic Design
Gamma: Blackmagic Design Film
ISO: 800
Highlight Recovery
✓ Apply LUT
LUT Source: Sidecar
Embedded
Sidecar

Color Temp: 5500
Tint: 3
Exposure: 0.00

Gamma Controls
Saturation: 1.00
Contrast: 1.00
Midpoint: 0.38
Highlight Rolloff: 1.00
Shadow Rolloff: 1.00
White Level: 1.00
Black Level: 0.00
Use Video Black Level

Example: For clips that have an embedded or sidecar 3DLUT available, an "Apply LUT" checkbox and "LUT Source" dropdown are shown.

Basic Types

enum

The enumerator type is represented differently on each platform, using the most appropriate system type:

Windows	`unsigned int`
macOS	`uint32_t`
Linux	`uint32_t`

uuid

The Universally unique identifier type is represented differently on each platform, using the most appropriate system type:

Windows	`GUID`
macOS	`CFUUIDBytes`
Linux	`GUID`

Boolean

A boolean is represented differently on each platform, using the most appropriate system type:

Windows	`BOOL`
macOS	`bool`
Linux	`bool`

int8_t

The signed 8 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`signed char`
macOS	`int8_t`
Linux	`int8_t`

uint8_t

The unsigned 8 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`unsigned char`
macOS	`uint8_t`
Linux	`uint8_t`

int16_t

The signed 16 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`short`
macOS	`int16_t`
Linux	`int16_t`

uint16_t

The unsigned 16 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`unsigned short`
macOS	`uint16_t`
Linux	`uint16_t`

int32_t

The signed 32 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`int`
macOS	`int32_t`
Linux	`int32_t`

uint32_t

The unsigned 32 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`unsigned int`
macOS	`uint32_t`
Linux	`uint32_t`

int64_t

The signed 64 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`long long`
macOS	`int64_t`
Linux	`int64_t`

uint64_t

The unsigned 64 bit integer type is represented differently on each platform, using the most appropriate system type:

Windows	`unsigned long long`
macOS	`uint64_t`
Linux	`uint64_t`

long

The long type is represented differently on each platform, using the most appropriate system type:

Windows	`LONG`
macOS	`long`
Linux	`long`

string

Strings are represented differently on each platform, using the most appropriate system type:

Windows	`BSTR`
macOS	`CFStringRef`
Linux	`const char*`

SafeArray

The SafeArray type is represented differently on each platform, using the most appropriate system type:

macOS	`SafeArray*`
Linux	`SafeArray*`

SafeArrayData

The SafeArrayData type is represented differently on each platform, using the most appropriate system type:

macOS	`CFMutableDataRef`
Linux	`void*`

Variant

The Variant type is represented differently on each platform, using the most appropriate system type:

Windows	`VARIANT`
macOS	`Variant`
Linux	`Variant`

BlackmagicRawVariantType

Variant types that may be stored as metadata

	Value		
Key	**macOS and Linux**	**Windows**	**Description**
blackmagicRawVariantTypeEmpty	0	VT_EMPTY	Undefined type
blackmagicRawVariantTypeU8	1	VT_UI1	Unsigned 8 bit integer
blackmagicRawVariantTypeS16	2	VT_I2	Signed 16 bit integer
blackmagicRawVariantTypeU16	3	VT_UI2	Unsigned 16 bit integer
blackmagicRawVariantTypeS32	4	VT_I4	Signed 32 bit integer
blackmagicRawVariantTypeU32	5	VT_UI4	Unsigned 32 bit integer
blackmagicRawVariantTypeFloat32	6	VT_R4	Single precision 32 bit (IEEE 754) floating point number
blackmagicRawVariantTypeString	7	VT_BSTR	String variable
blackmagicRawVariantTypeSafeArray	8	VT_SAFEARRAY	Array variable

BlackmagicRawResourceType

Used in IBlackmagicRawResourceManager

Key	**Value**	**Description**
blackmagicRawResourceTypeBufferCPU	'cpub'	Page aligned CPU addressable memory
blackmagicRawResourceTypeBufferMetal	'metb'	Metal MTLBuffer
blackmagicRawResourceTypeBufferCUDA	'cudb'	CUDA CUdeviceptr device pointer
blackmagicRawResourceTypeBufferOpenCL	'oclb'	OpenCL cl_mem buffer object

BlackmagicRawResourceFormat

Used for resource allocation

Key	Value	Description
blackmagicRawResourceFormatRGBAU8	‘rgba'	Unsigned 8bit interleaved RGBA
blackmagicRawResourceFormatBGRAU8	‘bgra’	Unsigned 8bit interleaved BGRA
blackmagicRawResourceFormatRGBU16	'16il'	Unsigned 16bit interleaved RGB
blackmagicRawResourceFormatRGBU16Planar	‘16pl'	Unsigned 16bit planar RGB
blackmagicRawResourceFormatRGBF32	‘f32s'	Floating point interleaved RGB
blackmagicRawResourceFormatRGBF32Planar	‘f32p'	Floating point planar RGB
blackmagicRawResourceFormatBGRAF32	‘f32a’	Floating point interleaved BGRA

BlackmagicRawResourceUsage

Used in IBlackmagicRawResourceManager

Key	Value	Description
blackmagicRawResourceUsageReadCPUWriteCPU	‘rcwc'	CPU readable and writable memory
blackmagicRawResourceUsageReadGPUWriteGPU	‘rgwg'	GPU readable and writable memory
blackmagicRawResourceUsageReadGPUWriteCPU	‘rgwc'	GPU readable, CPU writable memory
blackmagicRawResourceUsageReadCPUWriteGPU	‘rcwg'	CPU readable, GPU writable memory

BlackmagicRawPipeline

Used in IBlackmagicRawConfiguration. Each pipeline has different mappings to context/commandQueue

Key	Value	Description
blackmagicRawPipelineCPU	‘cpub'	None
blackmagicRawPipelineCUDA	‘cuda'	CUDA pipeline, context/commandQueue map to CUcontext/CUstream
blackmagicRawPipelineMetal	‘metl'	Metal pipeline, context/commandQueue map to nil/MTLCommandQueue
blackmagicRawPipelineOpenCL	‘opcl'	OpenCL pipeline, context/commandQueue map to cl_context/cl_command_queue

BlackmagicRawInstructionSet

Used in IBlackmagicRawConfiguration

Key	Value	Description
blackmagicRawInstructionSetSSE41	‘se41'	SSE 4.1 CPU Instruction Set
blackmagicRawInstructionSetAVX	‘avx_'	AVX CPU Instruction Set
blackmagicRawInstructionSetAVX2	‘avx2'	AVX2 CPU Instruction Set

BlackmagicRawAudioFormat

Used in IBlackmagicRawFileAudio

Key	Value	Description
blackmagicRawAudioFormatPCMLittleEndian	‘pcml’	PCM little endian audio

BlackmagicRawResolutionScale

Used in IBlackmagicRawFrame

Key	Value	Description
blackmagicRawResolutionScaleFull	‘full'	Full Resolution
blackmagicRawResolutionScaleHalf	‘half'	Half height and width
blackmagicRawResolutionScaleQuarter	‘qrtr'	Quarter height and width
blackmagicRawResolutionScaleEighth	‘eith'	Eighth height and width
blackmagicRawResolutionScaleFullUpsideDown	‘lluf’	Full Resolution (renders upside-down)
blackmagicRawResolutionScaleHalfUpsideDown	‘flah’	Half height and width (renders upside-down)
blackmagicRawResolutionScaleQuarterUpsideDown	‘rtrq’	Quarter height and width (renders upside-down)
blackmagicRawResolutionScaleEighthUpsideDown	‘htie’	Eighth height and width (renders upside-down)

BlackmagicRawClipProcessingAttribute

Variant types that may be stored as metadata

Key	Value	Description
blackmagicRawClipProcessingAttributeColorScienceGen	‘csgn'	Blackmagic Color Science generation
blackmagicRawClipProcessingAttributeGamma	‘gama'	The gamma curve
blackmagicRawClipProcessingAttributeGamut	‘gamt'	The color gamut
blackmagicRawClipProcessingAttributeToneCurveContrast	‘tcon'	Contrast used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeToneCurveSaturation	‘tsat'	Saturation used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeToneCurveMidpoint	‘tmid'	Midpoint used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeToneCurveHighlights	‘thih'	Highlight rolloff used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeToneCurveShadows	‘tsha'	Shadow rolloff used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeToneCurveVideoBlackLevel	‘tvbl'	VideoBlackLevel used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeToneCurveBlackLevel	‘tblk'	BlackLevel used in Blackmagic Design Custom Gamma

Key	Value	Description
blackmagicRawClipProcessingAttributeToneCurveWhiteLevel	‘twit'	WhiteLevel used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeHighlightRecovery	‘hlry'	Is highlight recovery enabled
blackmagicRawClipProcessingAttributeAnalogGainIsConstant	‘agic’	Is analog gain constant throughout the clip
blackmagicRawClipProcessingAttributeAnalogGain	‘gain’	Analog gain for entire clip if analog gain is constant, otherwise analog gain of the first frame
blackmagicRawClipProcessingAttributePost3DLUTMode	'lutm'	Is the Post 3D LUT being applied embedded, sidecar or disabled
blackmagicRawClipProcessingAttributeEmbeddedPost3DLUTName	'emln'	Name of embedded 3D LUT
blackmagicRawClipProcessingAttributeEmbeddedPost3DLUTTitle	'emlt'	Title of embedded 3D LUT
blackmagicRawClipProcessingAttributeEmbeddedPost3DLUTSize	'emls'	Size of embedded 3D LUT
blackmagicRawClipProcessingAttributeEmbeddedPost3DLUTData	'emld'	Float array of embedded 3D LUT data
blackmagicRawClipProcessingAttributeSidecarPost3DLUTName	'scln'	Name of sidecar 3D LUT
blackmagicRawClipProcessingAttributeSidecarPost3DLUTTitle	'sclt'	Title of sidecar 3D LUT
blackmagicRawClipProcessingAttributeSidecarPost3DLUTSize	'scls'	Size of sidecar 3D LUT
blackmagicRawClipProcessingAttributeSidecarPost3DLUTData	'scld'	Float array of sidecar 3D LUT data
blackmagicRawClipProcessingAttributeGamutCompressionEnable	‘gace’	Enable gamut compression

BlackmagicRawFrameProcessingAttribute

Variant types that may be stored as metadata

Key	Value	Description
blackmagicRawFrameProcessingAttributeWhiteBalanceKelvin	‘wbkv'	The white balance Kelvin value
blackmagicRawFrameProcessingAttributeWhiteBalanceTint	‘wbtn'	The white balance Tint value
blackmagicRawFrameProcessingAttributeExposure	‘expo'	The linear exposure adjustment value (in stops)
blackmagicRawFrameProcessingAttributeISO	‘fiso'	The ISO gamma curve
blackmagicRawFrameProcessingAttributeAnalogGain	‘agpf’	Analog Gain per-frame value, cannot be changed

BlackmagicRawInterop

Key	Value	Description
blackmagicRawInteropNone	'none'	None
blackmagicRawInteropOpenGL	'opgl'	None

Interface Reference

IBlackmagicRaw Interface

Each codec interface will have its own memory storage and decoder. When decoding multiple clips via one codec, first in first out ordering will apply

Related Interfaces

Interface	Interface ID
IBlackmagicRawClip	IID_IBlackmagicRawClip
IBlackmagicRawConfiguration	IID_IBlackmagicRawConfiguration
IBlackmagicRawConfigurationEx	IID_IBlackmagicRawConfigurationEx
IBlackmagicRawManualDecoderFlow1	IID_IBlackmagicRawManualDecoderFlow1
IBlackmagicRawManualDecoderFlow2	IID_IBlackmagicRawManualDecoderFlow2
IBlackmagicRawToneCurve	IID_IBlackmagicRawToneCurve
IBlackmagicRawFactory	IID_IBlackmagicRawFactory

Public Member Functions	
Method	**Description**
`OpenClip`	Opens a clip
`SetCallback`	Registers a callback with the codec object
`PreparePipeline`	Asynchronously prepares the current pipeline for decoding, calling the registered callback's PreparePipelineComplete() method upon completion. This reduces the potential performance impact of decoding the first frame due to on-demand GPU kernel compilation.
`PreparePipelineForDevice`	Asynchronously prepares the current pipeline for decoding, calling the registered callback's PreparePipelineComplete() method upon completion. This reduces the potential performance impact of decoding the first frame due to on-demand GPU kernel compilation.
`FlushJobs`	Blocking call which will only return once all jobs have been completed

IBlackmagicRaw::OpenClip method

Opens a clip

Syntax

```
HRESULT OpenClip (string fileName,IBlackmagicRawClip** clip)
```

Parameters

Name	Direction	Description
fileName	in	File name on disk of clip to open
clip	out	Returned object with opened clip

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when clip is NULL, E_INVALIDARG is returned when fileName is invalid, E_FAIL is returned if the clip failed to open.

IBlackmagicRaw::SetCallback method

Registers a callback with the codec object

Syntax

```
HRESULT SetCallback (IBlackmagicRawCallback* callback)
```

Parameters

Name	Direction	Description
callback	in	your callback object

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRaw::PreparePipeline method

Asynchronously prepares the current pipeline for decoding, calling the registered callback's **PreparePipelineComplete()** method upon completion. This reduces the potential performance impact of decoding the first frame due to on-demand GPU kernel compilation.

Syntax

```
HRESULT PreparePipeline (void* pipelineContext,
                         void* pipelineCommandQueue,
                         void* userData)
```

Parameters

Name	Direction	Description
pipeline	in	Pipeline for which to prepare. This must be the same as the current pipeline and is provided for validation purposes
pipelineContext	in	Context to use for preparation. For CPU/CUDA/Metal/OpenCL, this maps to null/CUcontext/null/cl_context
pipelineCommandQueue	in	Command queue to use for preparation. For CPU/CUDA/Metal/OpenCL, this maps to null/CUstream/MTLCommandQueue/cl_command_queue
userData	in	User data to pass through to the callback's PreparePipelineComplete() method

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRaw::PreparePipelineForDevice method

Asynchronously prepares the current pipeline for decoding, calling the registered callback's PreparePipelineComplete() method upon completion. This reduces the potential performance impact of decoding the first frame due to on-demand GPU kernel compilation.

Syntax

```
HRESULT PreparePipelineForDevice (IBlackmagicRawPipelineDevice*
                                  pipelineDevice,
                                  void* userData)
```

Parameters

Name	Direction	Description
pipelineDevice	in	The device to use for preparation
userData	in	User data to pass through to the callback's PreparePipelineComplete() method

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRaw::FlushJobs method

Blocking call which will only return once all jobs have been completed

Syntax

```
HRESULT FlushJobs()
```

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRawFactory Interface

Use this to create one or more Codec objects

Public Member Functions	
Method	**Description**
`CreateCodec`	Create a codec from the factory
`CreatePipelineIterator`	Create a pipeline iterator from the factory
`CreatePipelineDeviceIterator`	Create a pipeline device iterator from the factory

IBlackmagicRawFactory::CreateCodec method

Create a codec from the factory

Syntax

```
HRESULT CreateCodec (IBlackmagicRaw** codec)
```

Parameters

Name	Direction	Description
codec	out	Returned codec object

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when codec is NULL, E_FAIL is returned if the codec failed to create.

IBlackmagicRawFactory::CreatePipelineIterator method

Create a pipeline iterator from the factory

Syntax

```
HRESULT CreatePipelineIterator (BlackmagicRawInterop interop,
                                IBlackmagicRawPipelineIterator**
                                pipelineIterator)
```

Parameters

Name	Direction	Description
interop	in	Interoperability (with other APIs) required from the pipeline
pipelineIterator	out	The created pipeline iterator

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when pipelineIterator is NULL.

IBlackmagicRawFactory::CreatePipelineDeviceIterator method

Create a pipeline device iterator from the factory

Syntax

```
HRESULT CreatePipelineDeviceIterator (BlackmagicRawPipeline pipeline,
                                      BlackmagicRawInterop interop,
                                      IBlackmagicRawPipelineDeviceIterator**
                                      deviceIterator)
```

Parameters

Name	Direction	Description
pipeline	in	The pipeline from which to query the available devices
interop	in	Interoperability (with other APIs) required from the pipeline and devices
deviceIterator	out	The created pipeline device iterator

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when deviceIterator is NULL.

IBlackmagicRawPipelineIterator Interface

Use this to determine pipelines available for use on the system

Public Member Functions	
Method	**Description**
`Next`	Step to next pipeline entry. S_FALSE is returned when called on last entry
`GetName`	Get the name of the pipeline
`GetInterop`	Get the interoperability of the pipeline
`GetPipeline`	Get the pipeline

IBlackmagicRawPipelineIterator::Next method

Step to next pipeline entry. S_FALSE is returned when called on last entry

Syntax

```
HRESULT Next()
```

Return Values

If the method succeeds, the return value is S_OK or S_FALSE. S_FALSE is returned when **Next()** is called on the last element. E_FAIL is returned when **Next()** is called after the last element.

IBlackmagicRawPipelineIterator::GetName method

Get the name of the pipeline

Syntax

```
HRESULT GetName (string* pipelineName)
```

Parameters

Name	Direction	Description
pipelineName	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **pipelineName** is NULL.

IBlackmagicRawPipelineIterator::GetInterop method

Get the interoperability of the pipeline

Syntax

```
HRESULT GetInterop (BlackmagicRawInterop* interop)
```

Parameters

Name	Direction	Description
interop	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when interop is NULL.

IBlackmagicRawPipelineIterator::GetPipeline method

Get the pipeline

Syntax

```
HRESULT GetPipeline (BlackmagicRawPipeline* pipeline)
```

Parameters

Name	Direction	Description
pipeline	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when pipeline is NULL.

IBlackmagicRawPipelineDeviceIterator Interface

Use this to determine pipeline devices available for use on the system

Public Member Functions	
Method	**Description**
`Next`	Step to next device entry, will return S_FALSE when called on last entry
`GetPipeline`	Get the pipeline
`GetInterop`	Get the interoperability of the device's pipeline
`CreateDevice`	Create the pipeline device (container for context and command queue)

IBlackmagicRawPipelineDeviceIterator::Next method

Step to next device entry, will return S_FALSE when called on last entry

Syntax

```
HRESULT Next()
```

Return Values

If the method succeeds, the return value is S_OK or S_FALSE. S_FALSE is returned when **Next()** is called on the last element. E_FAIL is returned when **Next()** is called after the last element.

IBlackmagicRawPipelineDeviceIterator::GetPipeline method

Get the pipeline

Syntax

```
HRESULT GetPipeline (BlackmagicRawPipeline* pipeline)
```

Parameters

Name	Direction	Description
pipeline	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when pipeline is NULL.

IBlackmagicRawPipelineDeviceIterator::GetInterop method

Get the interoperability of the device's pipeline

Syntax

```
HRESULT GetInterop (BlackmagicRawInterop* interop)
```

Parameters

Name	Direction	Description
interop	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when interop is NULL.

IBlackmagicRawPipelineDeviceIterator::CreateDevice method

Create the pipeline device (container for context and command queue)

Syntax

```
HRESULT CreateDevice (IBlackmagicRawPipelineDevice** pipelineDevice)
```

Parameters

Name	Direction	Description
pipelineDevice	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when pipelineDevice is NULL.

IBlackmagicRawOpenGLInteropHelper Interface

Public Member Functions	
Method	**Description**
`GetPreferredResourceFormat`	Gets the preferred resource format for interaction between the device and OpenGL
`SetImage`	Copies the processed image into an OpenGL texture

IBlackmagicRawOpenGLInteropHelper::GetPreferredResourceFormat method

Gets the preferred resource format for interaction between the device and OpenGL

Syntax

```
HRESULT GetPreferredResourceFormat (BlackmagicRawResourceFormat* preferredFormat)
```

Parameters

Name	Direction	Description
preferredFormat	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when preferredFormat is NULL.

IBlackmagicRawOpenGLInteropHelper::SetImage method

Copies the processed image into an OpenGL texture

Syntax

```
HRESULT SetImage (IBlackmagicRawProcessedImage* processedImage,
                  uint32_t* openGLTextureName,
                  int32_t* openGLTextureTarget)
```

Parameters

Name	Direction	Description
processedImage	in	
openGLTextureName	out	name of OpenGL texture containing image
openGLTextureTarget	out	OpenGL target of texture containing image, typically GL_TEXTURE or GL_TEXTURE_RECTANGLE

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when processedImage is NULL.

IBlackmagicRawPipelineDevice Interface

A device is essentially a container for a context and command queue associated with a pipeline. This object is provided so the user need not deal directly with the underlying compute API in order to provide context and command queue to the codec configuration. As such, the device instance MUST outlive that of the codec instance on which the device is used.

Public Member Functions	
Method	**Description**
`SetBestInstructionSet`	Sets the CPU instruction set of the device to be that representing the best capabilities of the system
`SetInstructionSet`	Sets the CPU instruction set to use for the device
`GetInstructionSet`	Gets the CPU instruction set of the device
`GetIndex`	Gets the index of the device in the pipeline's device list. This is typically used to differentiate devices in multi-GPU configurations.
`GetName`	Gets the name of the device
`GetInterop`	Gets the API interoperability of the device
`GetPipeline`	Gets the pipeline configuration information associated with the device. These parameters may be provided to IBlackmagicRawConfiguration::SetPipeline. IBlackmagicRawConfiguration::SetFromDevice may be a better option.
`GetPipelineName`	Gets the name of the pipeline associated with the device
`GetOpenGLInteropHelper`	Creates an instance of a helper to get the results of a processed image as an OpenGL texture

IBlackmagicRawPipelineDevice::SetBestInstructionSet method

Sets the CPU instruction set of the device to be that representing the best capabilities of the system

Syntax

```
HRESULT SetBestInstructionSet()
```

Return Values

This method returns S_OK.

IBlackmagicRawPipelineDevice::SetInstructionSet method

Sets the CPU instruction set to use for the device

Syntax

```
HRESULT SetInstructionSet (BlackmagicRawInstructionSet instructionSet)
```

Parameters

Name	Direction	Description
instructionSet	in	–

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when **instructionSet** is not a valid **BlackmagicRawInstructionSet** enumeration value. E_FAIL is returned if the user's CPU does not support the specified instruction set.

IBlackmagicRawPipelineDevice::GetInstructionSet method

Gets the CPU instruction set of the device

Syntax

```
HRESULT GetInstructionSet (BlackmagicRawInstructionSet* instructionSet)
```

Parameters

Name	Direction	Description
instructionSet	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when instructionSet is NULL.

IBlackmagicRawPipelineDevice::GetIndex method

Gets the index of the device in the pipeline's device list. This is typically used to differentiate devices in multi-GPU configurations.

Syntax

```
HRESULT GetIndex (uint32_t* deviceIndex)
```

Parameters

Name	Direction	Description
deviceIndex	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when deviceIndex is NULL.

IBlackmagicRawPipelineDevice::GetName method

Gets the name of the device

Syntax

```
HRESULT GetName (string* deviceName)
```

Parameters

Name	Direction	Description
deviceName	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when deviceName is NULL.

IBlackmagicRawPipelineDevice::GetInterop method

Gets the API interoperability of the device

Syntax

```
HRESULT GetInterop (BlackmagicRawInterop* interop)
```

Parameters

Name	Direction	Description
interop	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when interop is NULL.

IBlackmagicRawPipelineDevice::GetPipeline method

Gets the pipeline configuration information associated with the device. These parameters may be provided to IBlackmagicRawConfiguration::SetPipeline.
IBlackmagicRawConfiguration::SetFromDevice may be a better option.

Syntax

```
HRESULT GetPipeline (BlackmagicRawPipeline* pipeline,
                     void** context,
                     void** commandQueue)
```

Parameters

Name	Direction	Description
pipeline	out	–
context	out	–
commandQueue	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when any of pipeline, context and commandQueue is NULL.

IBlackmagicRawPipelineDevice::GetPipelineName method

Gets the name of the pipeline associated with the device

Syntax

```
HRESULT GetPipelineName (string* pipelineName)
```

Parameters

Name	Direction	Description
pipelineName	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **pipelineName** is NULL.

IBlackmagicRawPipelineDevice::GetOpenGLInteropHelper method

Creates an instance of a helper to get the results of a processed image as an OpenGL texture

Syntax

```
HRESULT GetOpenGLInteropHelper (IBlackmagicRawOpenGLInteropHelper**
                                 interopHelper)
```

Parameters

Name	Direction	Description
interopHelper	out	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when interopHelper is NULL.

IBlackmagicRawToneCurve Interface

If desired, the user application can cache these results

Related Interfaces

Public Member Functions	
Method	**Description**
`GetToneCurve`	Query tone curve parameters for a specific camera and gamma. These are only currently available on Gamut: Blackmagic Design, Gamma: Blackmagic Design Film, Blackmagic Design Extended Video, Blackmagic Design Custom. Note: Custom gamma can define a tone curve per clip, see BlackmagicRawClipProcessingAttributes::GetToneCurveForCustomGamma()
`EvaluateToneCurve`	Evaluates tone curve, returned buffer can be used to visualise curve

IBlackmagicRawToneCurve::GetToneCurve method

Query tone curve parameters for a specific camera and gamma. These are only currently available on Gamut: Blackmagic Design, Gamma: Blackmagic Design Film, Blackmagic Design Extended Video, Blackmagic Design Custom.

Note: Custom gamma can define a tone curve per clip, see BlackmagicRawClipProcessingAttributes::GetToneCurveForCustomGamma()

Syntax

```
HRESULT GetToneCurve (string cameraType,
                      string gamma,
                      uint16_t gen,
                      float* contrast,
                      float* saturation,
                      float* midpoint,
                      float* highlights,
                      float* shadows,
                      float* blackLevel,
                      float* whiteLevel,
                      uint16_t* videoBlackLevel)
```

Parameters

Name	Direction	Description
cameraType	in	Type of camera, you can query this from **IBlackmagicRawClip::GetCameraType()**
gamma	in	String value of gamma to use
gen	in	Color science gen
contrast	out	Contrast of tonecurve
saturation	out	Saturation of tonecurve
midpoint	out	Midpoint of tonecurve
highlights	out	Control the highlights in the tonecurve
shadows	out	Control the shadows in the tonecurve
blackLevel	out	Black level in the tonecurve
whiteLevel	out	White level in the tonecurve
videoBlackLevel	out	Whether there is a black level pedestal applied

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when any of contrast, saturation, midpoint, highlights, shadows or videoBlackLevel are NULL. E_INVALIDARG is returned when the provided cameraType / gamma / gen combination is invalid.

IBlackmagicRawToneCurve::EvaluateToneCurve method

Evaluates tone curve, returned buffer can be used to visualise curve

Syntax

```
HRESULT EvaluateToneCurve (string cameraType,
                           uint16_t gen,
                           float contrast,
                           float saturation,
                           float midpoint,
                           float highlights,
                           float shadows,
                           float blackLevel,
                           float whiteLevel,
                           uint16_t videoBlackLevel,
                           float* array,
                           uint32_t arrayElementCount)
```

Parameters

Name	Direction	Description
cameraType	in	Type of camera, you can query this from **IBlackmagicRawClip::GetCameraType()**
gen	in	Color science gen
contrast	in	Contrast of tonecurve
saturation	in	Saturation of tonecurve
midpoint	in	Midpoint of tonecurve
highlights	in	Highlights of tonecurve
shadows	in	Shadows of tonecurve
blackLevel	in	Black level of tonecurve
whiteLevel	in	White level of tonecurve
videoBlackLevel	in	Do we apply a black level pedestal
array	out	Array to write the evaluated tonecurve in to
arrayElementCount	in	Size of array being provided

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when arrayOut is NULL. E_INVALIDARG is returned when arrayOutElementCount is 0 or the cameraType / bmdgen combination provided is invalid.

IBlackmagicRawConfiguration Interface

The configuration properties are read when the first call to OpenClip() occurs. After this configuration properties should not be changed, and changes will be ignored.

Related Interfaces

Interface	Interface ID
IBlackmagicRaw	IID_IBlackmagicRaw

Public Member Functions	
Method	**Description**
`SetPipeline`	Set pipeline to use for decoding, see BlackmagicRawPipeline
`GetPipeline`	Get pipeline used for decoding, see BlackmagicRawPipeline
`IsPipelineSupported`	Determine if a pipeline is supported by this machine. This will verify relevant hardware / DLLs are installed
`SetCPUThreads`	Sets the number of CPU threads to use while decoding. Defaults to number of hardware threads available on system
`GetCPUThreads`	Gets the number of CPU threads to use while decoding
`GetMaxCPUThreadCount`	Query the number of hardware threads available on system
`SetWriteMetadataPerFrame`	Sets if per-frame metadata will be written to only the relevant frame.
`GetWriteMetadataPerFrame`	Gets if the per-frame metadata will be written to only the relevant frame
`SetFromDevice`	Equivalent to querying the device for instruction set, pipeline, context and command queue then calling SetInstructionSet and SetPipeline

IBlackmagicRawConfiguration::SetPipeline method

Set pipeline to use for decoding, see BlackmagicRawPipeline

Syntax

```
HRESULT SetPipeline (BlackmagicRawPipeline pipeline,
                     void* pipelineContext,
                     void* pipelineCommandQueue)
```

Parameters

Name	Direction	Description
pipeline	in	Set pipeline before allocating resources, as changing pipeline will cause the default resource manager to be re-created
pipelineContext	in	Set context to use. For CPU/CUDA/Metal/OpenCL maps to null/CUcontext/null/cl_context
pipelineCommandQueue	in	Sets commandQueue to use. For CPU/CUDA/Metal/OpenCL maps to null/CUstream/MTLCommandQueue/cl_command_queue

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned when the pipeline failed to initialise.

IBlackmagicRawConfiguration::GetPipeline method

Get pipeline used for decoding, see BlackmagicRawPipeline

Syntax

```
HRESULT GetPipeline (BlackmagicRawPipeline* pipeline,
                     void** pipelineContextOut,
                     void** pipelineCommandQueueOut)
```

Parameters

Name	Direction	Description
pipeline	out	returns the pipeline used
pipelineContextOut	out	Returns context applied. For CPU/CUDA/Metal/OpenCL maps to null/CUcontext/null/cl_context
pipelineCommandQueueOut	out	Returns commandQueue applied. For CPU/CUDA/Metal/OpenCL maps to null/CUstream/MTLCommandQueue/cl_command_queue

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRawConfiguration::IsPipelineSupported method

Determine if a pipeline is supported by this machine. This will verify relevant hardware / DLLs are installed

Syntax

```
HRESULT IsPipelineSupported (BlackmagicRawPipeline pipeline,
                             Boolean* pipelineSupported)
```

Parameters

Name	Direction	Description
pipeline	in	Type of pipeline to query
pipelineSupported	out	Returned result

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when clip is NULL, E_INVALIDARG is returned when pipeline is invalid

IBlackmagicRawConfiguration::SetCPUThreads method

Sets the number of CPU threads to use while decoding. Defaults to number of hardware threads available on system

Syntax

```
HRESULT SetCPUThreads (uint32_t threadCount)
```

Parameters

Name	Direction	Description
threadCount	in	Thread count to utilise, setting to 0 will default to number of hardware threads available on system

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRawConfiguration::GetCPUThreads method

Gets the number of CPU threads to use while decoding

Syntax

```
HRESULT GetCPUThreads (uint32_t* threadCount)
```

Parameters

Name	Direction	Description
threadCount	out	Returned thread count

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when threadCount is NULL.

IBlackmagicRawConfiguration::GetMaxCPUThreadCount method

Query the number of hardware threads available on system

Syntax

```
HRESULT GetMaxCPUThreadCount (uint32_t* threadCount)
```

Parameters

Name	Direction	Description
threadCount	out	Returned thread count

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when maxCPUThreadCount is NULL.

IBlackmagicRawConfiguration::SetWriteMetadataPerFrame method

Sets if per-frame metadata will be written to only the relevant frame.

Syntax

```
HRESULT SetWriteMetadataPerFrame (Boolean writePerFrame)
```

Parameters

Name	Direction	Description
writePerFrame	in	if true, frame metadata will be written to only the relevant frame, if false, setting frame metadata will set to all frames at once

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRawConfiguration::GetWriteMetadataPerFrame method

Gets if the per-frame metadata will be written to only the relevant frame

Syntax

```
HRESULT GetWriteMetadataPerFrame (Boolean* writePerFrame)
```

Parameters

Name	Direction	Description
writePerFrame	out	if true, frame metadata will be written to only the relevant frame, if false, setting frame metadata will set to all frames at once

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when writePerFrame is NULL.

IBlackmagicRawConfiguration::SetFromDevice method

Equivalent to querying the device for instruction set, pipeline, context and command queue then calling SetInstructionSet and SetPipeline

Syntax

```
HRESULT SetFromDevice (IBlackmagicRawPipelineDevice* pipelineDevice)
```

Parameters

Name	Direction	Description
pipelineDevice	in	–

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when the pipelineDevice is NULL.

IBlackmagicRawConfigurationEx Interface

Extended Configuration for Codec Object

Related Interfaces

Interface	Interface ID
IBlackmagicRaw	IID_IBlackmagicRaw

Public Member Functions	
Method	**Description**
`GetResourceManager`	Get the current resource manager
`SetResourceManager`	Set the current resource manager, this allows the user to provide a custom resource manager
`GetInstructionSet`	Get the CPU instruction set used by the decoder
`SetInstructionSet`	Set the CPU instruction set used by the decoder

IBlackmagicRawConfigurationEx::GetResourceManager method

Get the current resource manager

Syntax

```
HRESULT GetResourceManager (IBlackmagicRawResourceManager** resourceManager)
```

Parameters

Name	Direction	Description
resourceManager	out	Returned resource manager

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resourceManager is NULL.

IBlackmagicRawConfigurationEx::SetResourceManager method

Set the current resource manager, this allows the user to provide a custom resource manager

Syntax

```
HRESULT SetResourceManager (IBlackmagicRawResourceManager* resourceManager)
```

Parameters

Name	Direction	Description
resourceManager	in	setting null will restore the default resource manager

Return Values

If the method succeeds, the return value is S_OK. E_FAIL can occur when setting the a NULL resource manager and the default resource manager failed to create.

IBlackmagicRawConfigurationEx::GetInstructionSet method

Get the CPU instruction set used by the decoder

Syntax

```
HRESULT GetInstructionSet (BlackmagicRawInstructionSet* instructionSet)
```

Parameters

Name	Direction	Description
instructionSet	out	Returned instruction set

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when instructionSet is NULL.

IBlackmagicRawConfigurationEx::SetInstructionSet method

Set the CPU instruction set used by the decoder

Syntax

```
HRESULT SetInstructionSet (BlackmagicRawInstructionSet instructionSet)
```

Parameters

Name	Direction	Description
instructionSet	in	the instruction set to use

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when instructionSet is invalid. E_FAIL is returned when the system does not support the provided instruction set.

IBlackmagicRawResourceManager Interface

Using this interface the user can create their own Resource manager to allow ownership over resource allocations. An internal resource manager that implements this is interface is provided by default.

Public Member Functions	
Method	**Description**
`CreateResource`	Called when a new resource is created
`ReleaseResource`	Release a resource
`CopyResource`	Copy a resource
`GetResourceHostPointer`	Obtains a pointer to a resource's host addressable memory

IBlackmagicRawResourceManager::CreateResource method

Called when a new resource is created

Syntax

```
HRESULT CreateResource (void* context,
                        void* commandQueue,
                        uint32_t sizeBytes,
                        BlackmagicRawResourceType type,
                        BlackmagicRawResourceUsage usage,
                        void** resource)
```

Parameters

Name	Direction	Description
context	in	Context on which to create the resource
commandQueue	in	Command Queue on which to create the resource
sizeBytes	in	Size (in bytes) of the resource to create
type	in	Type of resource to create
usage	in	Usage of resource to create
resource	out	Return the created resource

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resource is NULL. E_INVALIDARG is returned when type is invalid or does not match the current pipeline. E_OUTOFMEMORY is returned if the allocation failed.

IBlackmagicRawResourceManager::ReleaseResource method

Release a resource

Syntax

```
HRESULT ReleaseResource (void* context,
                         void* commandQueue,
                         void* resource,
                         BlackmagicRawResourceType type)
```

Parameters

Name	Direction	Description
context	in	Context the resource was created on
commandQueue	in	CommandQueue the resource was created on
resource	in	Resource to release
type	in	Type of resource we are releasing

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when type is invalid or does not match the current pipeline. E_UNEXPECTED is returned if an unexpected error occurs.

IBlackmagicRawResourceManager::CopyResource method

Copy a resource

Syntax

```
HRESULT CopyResource (void* context,
                      void* commandQueue,
                      void* source,
                      BlackmagicRawResourceType sourceType,
                      void* destination,
                      BlackmagicRawResourceType destinationType,
                      uint32_t sizeBytes,
                      Boolean copyAsync)
```

Parameters

Name	Direction	Description
context	in	Context the resource was created on
commandQueue	in	CommandQueue the resource was created on
source	in	Source resource to copy
sourceType	in	Type of resource to copy from
destination	in	Destination resource of the copy
destinationType	in	Type of resource to copy to
sizeBytes	in	Size (in bytes) of the resource to copy

Name	Direction	Description
copyAsync	in	if true, queue the copy to happen asynchronously (implying the source buffer MUST exist for the duration)

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when type is invalid or does not match the current pipeline. E_UNEXPECTED is returned if an unexpected error occurs.

IBlackmagicRawResourceManager::GetResourceHostPointer method

Obtains a pointer to a resource's host addressable memory

Syntax

```
HRESULT GetResourceHostPointer (void* context,
                                void* commandQueue,
                                void* resource,
                                BlackmagicRawResourceType resourceType,
                                void** hostPointer)
```

Parameters

Name	Direction	Description
context	in	Context the resource was created on
commandQueue	in	CommandQueue the resource was created on
resource	in	Resource to query
resourceType	in	Type of resource we are querying
hostPointer	out	Resultant host pointer of the resource

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when type is invalid or does not match the current pipeline.

IBlackmagicRawMetadataIterator Interface

Iterating metadata

Public Member Functions	
Method	**Description**
`Next`	Step to next metadata entry, will return S_FALSE when called on last entry
`GetKey`	Query key name of this metadata entry
`GetData`	Query data in this metadata entry

IBlackmagicRawMetadataIterator::Next method

Step to next metadata entry, will return S_FALSE when called on last entry

Syntax

```
HRESULT Next()
```

Return Values

If the method succeeds, the return value is S_OK or S_FALSE. S_FALSE is returned when **Next()** is called on the last element. E_FAIL is returned when **Next()** is called after the last element.

IBlackmagicRawMetadataIterator::GetKey method

Query key name of this metadata entry

Syntax

```
HRESULT GetKey (string* key)
```

Parameters

Name	Direction	Description
key	out	Name of key

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when key is NULL, E_FAIL is returned if the iterator has already stepped past the last element

IBlackmagicRawMetadataIterator::GetData method

Query data in this metadata entry

Syntax

```
HRESULT GetData (Variant* data)
```

Parameters

Name	Direction	Description
data	out	Variant to store the data in

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when key is NULL, E_FAIL is returned if the iterator has already stepped past the last element

IBlackmagicRawClipProcessingAttributes Interface

Clip Processing attributes allows the user to adjust clip-level processing attributes

Related Interfaces

Interface	Interface ID
IBlackmagicRawPost3DLUT	IID_IBlackmagicRawPost3DLUT
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions	
Method	**Description**
`GetClipAttribute`	Get the attribute
`SetClipAttribute`	Set the attribute
`GetClipAttributeRange`	Get the clip processing attribute range for the specified attribute
`GetClipAttributeList`	Get the clip processing attribute value list for the specified attribute. The arrayElementCount may be queried first (with NULL array parameter) to allocate correct size. A subsequent call (with non-NULL array parameter) can be used to populate the array.

Public Member Functions	
Method	**Description**
`GetISOList`	Obtains a list of available ISOs (for the clip's analog gain) which is primarily intended for GUI presentation.
`GetPost3DLUT`	Get the active 3D LUT

IBlackmagicRawClipProcessingAttributes::GetClipAttribute method

Get the attribute

Syntax

```
HRESULT GetClipAttribute (BlackmagicRawClipProcessingAttribute attribute,
                          Variant* value)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to query
value	out	Variant to store the queried value in

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when attribute, E_POINTER is returned when value is NULL.

IBlackmagicRawClipProcessingAttributes::SetClipAttribute method

Set the attribute

Syntax

```
HRESULT SetClipAttribute(BlackmagicRawClipProcessingAttribute attribute,
                         Variant* value)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to set
value	in	Variant to set the value to

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when attribute or value is invalid. When changing the Sidecar 3D LUT parameters, it's possible to set an attribute that is valid but incompatible with the existing state (e.g. LUT Size = 33 when the current LUT Data is for a 17x17x17 cube). In this case, the method will return S_FALSE and temporarily disable the active sidecar LUT until a full set of valid parameters are specified.

IBlackmagicRawClipProcessingAttributes::GetClipAttributeRange method

Get the clip processing attribute range for the specified attribute

Syntax

```
HRESULT GetClipAttributeRange(BlackmagicRawClipProcessingAttribute attribute,
                              Variant* valueMin,
                              Variant* valueMax,
                              Boolean* isReadOnly)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to query
valueMin	out	Variant to store the data in
valueMax	out	Variant to store the data in
isReadOnly	out	Returned boolean indicating if this attribute can be modified. Serves as a hint that any corresponding GUI control shall be disabled.

IBlackmagicRawClipProcessingAttributes::GetClipAttributeList method

Get the clip processing attribute value list for the specified attribute. The arrayElementCount may be queried first (with NULL array parameter) to allocate correct size. A subsequent call (with non-NULL array parameter) can be used to populate the array.

Syntax

```
HRESULT GetClipAttributeList(BlackmagicRawClipProcessingAttribute attribute,
                             Variant* array,
                             uint32_t* arrayElementCount,
                             Boolean* isReadOnly)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to query
array	out	The array into which the results will be written. If nullptr is supplied then arrayElementCount will still be returned.
arrayElementCount	out	Array element count
isReadOnly	out	Returned boolean indicating if this attribute can be modified. Serves as a hint that any corresponding GUI control shall be disabled.

IBlackmagicRawClipProcessingAttributes::GetISOList method

Obtains a list of available ISOs (for the clip's analog gain) which is primarily intended for GUI presentation.

Syntax

```
HRESULT GetISOList(uint32_t* array,
                   uint32_t* arrayElementCount,
                   Boolean* isReadOnly)
```

Parameters

Name	Direction	Description
array	out	The array into which the results will be written. If nullptr is supplied then arrayElementCount will still be returned.
arrayElementCount	in, out	Array element count. Input value shall indicate the number of elements available in array. Output value indicates the number of elements populated in array.
isReadOnly	out	Returned boolean indicating if this attribute can be modified. Serves as an indication that any corresponding GUI control shall be disabled.

IBlackmagicRawClipProcessingAttributes::GetPost3DLUT method

Get the active 3D LUT

Syntax

```
HRESULT GetPost3DLUT (IBlackmagicRawPost3DLUT** lut)
```

Parameters

Name	Direction	Description
lut	out	Look up table (LUT) to query

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when lut is NULL.

IBlackmagicRawFrameProcessingAttributes Interface

Processing attributes which can change per frame

Related Interfaces

Interface	Interface ID
IBlackmagicRawFrame	IID_IBlackmagicRawFrame

Public Member Functions	
Method	**Description**
`GetFrameAttribute`	Get the attribute
`SetFrameAttribute`	Set the attribute
`GetFrameAttributeRange`	Get the frame processing attribute range for the specified attribute
`GetFrameAttributeList`	Get the frame processing attribute value list for the specified attribute. To query an ISO list use GetISOList().
`GetISOList`	Obtains a list of available ISOs (for the frame's analog gain) which is primarily intended for GUI presentation.

IBlackmagicRawFrameProcessingAttributes::GetFrameAttribute method

Get the attribute

Syntax

```
HRESULT GetFrameAttribute(BlackmagicRawFrameProcessingAttribute attribute,
                         Variant* value)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to query
value	out	Variant to store the queried value in

IBlackmagicRawFrameProcessingAttributes::SetFrameAttribute method

Set the attribute

Syntax

```
HRESULT SetFrameAttribute(BlackmagicRawFrameProcessingAttribute attribute,
                         Variant* value)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to set
value	in	Variant to set the attribute to

IBlackmagicRawFrameProcessingAttributes::GetFrameAttributeRange method

Get the frame processing attribute range for the specified attribute

Syntax

```
HRESULT GetFrameAttributeRange(BlackmagicRawFrameProcessingAttribute attribute,
                               Variant* valueMin,
                               Variant* valueMax,
                               Boolean* isReadOnly)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to query
valueMin	out	Variant to store the data in
valueMax	out	Variant to store the data in
isReadOnly	out	Returned boolean indicating if this attribute can be modified. Serves as a hint that any corresponding GUI control shall be disabled.

IBlackmagicRawFrameProcessingAttributes::GetFrameAttributeList method

Get the frame processing attribute value list for the specified attribute. To query an ISO list use **GetISOList**().

Syntax

```
HRESULT GetFrameAttributeList(BlackmagicRawFrameProcessingAttribute attribute,
                              Variant* array,
                              uint32_t* arrayElementCount,
                              Boolean* isReadOnly)
```

Parameters

Name	Direction	Description
attribute	in	Attribute to query
array	out	The array into which the results will be written. If nullptr is supplied then arrayElementCount will still be returned.
arrayElementCount	out	Array element count
isReadOnly	out	Returned boolean indicating if this attribute can be modified. Serves as a hint that any corresponding GUI control shall be disabled.

IBlackmagicRawFrameProcessingAttributes::GetISOList method

Obtains a list of available ISOs (for the frame's analog gain) which is primarily intended for GUI presentation.

Syntax

```
HRESULT GetISOList(uint32_t* array,
                   uint32_t* arrayElementCount,
                   Boolean* isReadOnly)
```

Parameters

Name	Direction	Description
array	out	The array into which the results will be written. If nullptr is supplied then arrayElementCount will still be returned.
arrayElementCount	in, out	Array element count. Input value shall indicate the number of elements available in array. Output value indicates the number of elements populated in array.
isReadOnly	out	Returned boolean indicating if this attribute can be modified. Serves as an indication that any corresponding GUI control shall be disabled.

IBlackmagicRawPost3DLUT Interface

3D Look up table (LUT) object. This object provides additional information about LUTs and gives user the ability to control the lifetime of the resource.

Public Member Functions	
Method	**Description**
`GetName`	Get the name of the 3D LUT
`GetTitle`	Get the title of the 3D LUT
`GetSize`	Get the size of the LUT. Eg, will return 17 for a 17x17x17 LUT.
`GetResourceGPU`	Get pointer to GPU resource the LUT is stored in
`GetResourceCPU`	Get pointer to CPU resource the LUT is stored in
`GetResourceSizeBytes`	Get size of resource in bytes

IBlackmagicRawPost3DLUT::GetName method

Get the name of the 3D LUT

Syntax

```
HRESULT GetName (string* name)
```

Parameters

Name	Direction	Description
name	out	Returned name

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when name is NULL.

IBlackmagicRawPost3DLUT::GetTitle method

Get the title of the 3D LUT

Syntax

```
HRESULT GetTitle (string* title)
```

Parameters

Name	Direction	Description
title	out	Returned title

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when title is NULL.

IBlackmagicRawPost3DLUT::GetSize method

Get the size of the LUT. Eg, will return 17 for a 17x17x17 LUT.

Syntax

```
HRESULT GetSize (uint32_t* size)
```

Parameters

Name	Direction	Description
size	out	Returned size in pixels

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when size is NULL.

IBlackmagicRawPost3DLUT::GetResourceGPU method

Get pointer to GPU resource the LUT is stored in

Syntax

```
HRESULT GetResourceGPU (void* context,
                        void* commandQueue,
                        BlackmagicRawResourceType* type,
                        void** resource)
```

Parameters

Name	Direction	Description
context	in	Context the resource should belong to. This will be API dependant, see BlackmagicRawPipeline for details
commandQueue	in	Command queue the resource should belong to. This will be API dependant, see BlackmagicRawPipeline for details
type	out	Returned type of resource
resource	out	This will differ per API. See BlackmagicRawResourceType for details

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resource or type is NULL. E_OUTOFMEMORY is returned when the resource is lazy created and the memory allocation failed.

IBlackmagicRawPost3DLUT::GetResourceCPU method

Get pointer to CPU resource the LUT is stored in

Syntax

```
HRESULT GetResourceCPU (void** resource)
```

Parameters

Name	Direction	Description
resource	out	CPU resource object

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resource is NULL.

IBlackmagicRawPost3DLUT::GetResourceSizeBytes method

Get size of resource in bytes

Syntax

```
HRESULT GetResourceSizeBytes (uint32_t* sizeBytes)
```

Parameters

Name	Direction	Description
sizeBytes	out	Returned size of resource in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when sizeBytes is NULL.

IBlackmagicRawProcessedImage Interface

This object is created by the API and provided via a ProcessComplete() callback.

Public Member Functions	
Method	**Description**
`GetWidth`	Get the width of the processed image
`GetHeight`	Get the height of the processed image
`GetResource`	Get pointer to resource the image is stored in
`GetResourceType`	Get type of resource, see BlackmagicRawResourceType
`GetResourceFormat`	Get format of resource, see BlackmagicRawResourceFormat
`GetResourceSizeBytes`	Get size of resource in bytes
`GetResourceContextAndCommandQueue`	Get context and command queue that the resource was created on

IBlackmagicRawProcessedImage::GetWidth method

Get the width of the processed image

Syntax

```
HRESULT GetWidth (uint32_t* width)
```

Parameters

Name	Direction	Description
width	out	Returned width in pixels

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when width is NULL.

IBlackmagicRawProcessedImage::GetHeight method

Get the height of the processed image

Syntax

```
HRESULT GetHeight (uint32_t* height)
```

Parameters

Name	Direction	Description
height	out	Returned height in pixels

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when height is NULL.

IBlackmagicRawProcessedImage::GetResource method

Get pointer to resource the image is stored in

Syntax

```
HRESULT GetResource (void** resource)
```

Parameters

Name	Direction	Description
resource	out	This will differ per API. See **BlackmagicRawResourceType** for details

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resource is NULL.

IBlackmagicRawProcessedImage::GetResourceType method

Get type of resource, see BlackmagicRawResourceType

Syntax

```
HRESULT GetResourceType (BlackmagicRawResourceType* type)
```

Parameters

Name	Direction	Description
type	out	Returned type of resource

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when type is NULL.

IBlackmagicRawProcessedImage::GetResourceFormat method

Get format of resource, see BlackmagicRawResourceFormat

Syntax

```
HRESULT GetResourceFormat (BlackmagicRawResourceFormat* format)
```

Parameters

Name	Direction	Description
format	out	Returned format of resource

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when format is NULL.

IBlackmagicRawProcessedImage::GetResourceSizeBytes method

Get size of resource in bytes

Syntax

```
HRESULT GetResourceSizeBytes (uint32_t* sizeBytes)
```

Parameters

Name	Direction	Description
sizeBytes	out	Returned size of resource in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when sizeBytes is NULL.

IBlackmagicRawProcessedImage::GetResourceContextAndCommandQueue method

Get context and command queue that the resource was created on

Syntax

```
HRESULT GetResourceContextAndCommandQueue (void** context,void** commandQueue)
```

Parameters

Name	Direction	Description
context	out	Returned context resource was created on, this native object will differ per API, see BlackmagicRawPipeline
commandQueue	out	Returned command queue resource was created on, this native object will differ per API, see BlackmagicRawPipeline

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when either context or commandQueue is NULL.

IBlackmagicRawJob Interface

This is the base object that is returned when any job is created with the SDK. Use this to control and identify jobs when callbacks occur.

Public Member Functions	
Method	**Description**
`Submit`	Submit the job to the decoder. This will insert the job in the decoders internal queue. From here the relevant callback (i.e. ProcessComplete()) will occur as soon as the job is completed. Note: When queuing on GPU decoders, this function will not return until the job has been submitted to the internal GPU API. So you can use GPU synchronization methods rather than waiting for the CPU callbacks.
`Abort`	Abort the job. This CAN fail if the job has already been started by the internal decoder.
`SetUserData`	Attach some generic userdata to the job object/
`GetUserData`	Retrieve previously attached generic userdata from the job object

IBlackmagicRawJob::Submit method

Submit the job to the decoder. This will insert the job in the decoders internal queue. From here the relevant callback (i.e. **ProcessComplete()**) will occur as soon as the job is completed.

Note: When queuing on GPU decoders, this function will not return until the job has been submitted to the internal GPU API. So you can use GPU synchronization methods rather than waiting for the CPU callbacks.

Syntax

```
HRESULT Submit()
```

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned if the job has already been started. E_OUTOFMEMORY can be returned if the operation required memory and the allocation failed.

IBlackmagicRawJob::Abort method

Abort the job. This CAN fail if the job has already been started by the internal decoder.

Syntax

```
HRESULT Abort()
```

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned if the job has already been aborted, or if the job cannot abort now (for example it may have been sent to GPU already)

IBlackmagicRawJob::SetUserData method

Attach some generic userdata to the job object/

Syntax

```
HRESULT SetUserData (void* userData)
```

Parameters

Name	Direction	Description
userData	in	Userdata to attach

Return Values

If the method succeeds, the return value is S_OK.

IBlackmagicRawJob::GetUserData method

Retrieve previously attached generic userdata from the job object

Syntax

```
HRESULT GetUserData (void** userData)
```

Parameters

Name	Direction	Description
userData	out	Userdata that was attached

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when userData is NULL.

IBlackmagicRawCallback Interface

Central callback object for entire codec. Jobs submitted to any clip created by this codec will have their results provided through these function calls

Public Member Functions	
Method	**Description**
`ReadComplete`	Called when a read has completed
`DecodeComplete`	Called when a decode has completed
`ProcessComplete`	Called when a process has completed
`TrimProgress`	Called as a Trim job is processed to provide status updates
`TrimComplete`	Called when a trim has completed
`SidecarMetadataParseWarning`	Called when a parse warning occured when reading a related .sidecar file. Note: Parse warnings are not fatal, the offending line will be ignored. When SaveSidecarFile() is next called, the offending line will be removed.
`SidecarMetadataParseError`	Called when a parse error occured when reading a related .sidecar file. Note: If a parse error occurs, the entire file is ignored. When SaveSidecarFile() file is next called, the entire file will be replaced.
`PreparePipelineComplete`	Called when preparation of the pipeline has completed

IBlackmagicRawCallback::ReadComplete method

Called when a read has completed

Syntax

```
void ReadComplete (IBlackmagicRawJob* job, HRESULT result,
                   IBlackmagicRawFrame* frame)
```

Parameters

Name	Direction	Description
job	in	Job created to perform the read, see CreateJobReadFrame()
result	in	Result of the job. If the job succeeded, the job result is S_OK. The job result is E_UNEXPECTED if a dropped frame was encountered.
frame	in	Frame created (will be null if the job failed)

IBlackmagicRawCallback::DecodeComplete method

Called when a decode has completed

Syntax

```
void DecodeComplete (IBlackmagicRawJob* job, HRESULT result)
```

Parameters

Name	Direction	Description
job	in	Job created to perform the decode, see CreateJobDecode(). Note: this function is only used with manual decoders
result	in	Result of the job

IBlackmagicRawCallback::ProcessComplete method

Called when a process has completed

Syntax

```
void ProcessComplete (IBlackmagicRawJob* job, HRESULT result,
                      IBlackmagicRawProcessedImage* processedImage)
```

Parameters

Name	Direction	Description
job	in	Job created to perform the process, see CreateJobDecodeAndProcess() or CreateJobProcess()
result	in	Result of the job
processedImage	in	Create processed frame. This contains the final image ready for display

IBlackmagicRawCallback::TrimProgress method

Called as a Trim job is processed to provide status updates

Syntax

```
void TrimProgress (IBlackmagicRawJob* job, float progress)
```

Parameters

Name	Direction	Description
job	in	Job created to perform the trim
progress	in	Progress [0, 1] which defines how the trim operation has progressed

IBlackmagicRawCallback::TrimComplete method

Called when a trim has completed

Syntax

```
void TrimComplete (IBlackmagicRawJob* job, HRESULT result)
```

Parameters

Name	Direction	Description
job	in	Job created to perform the trim
result	in	Result of the job

IBlackmagicRawCallback::SidecarMetadataParseWarning method

Called when a parse warning occured when reading a related .sidecar file.

Note: Parse warnings are not fatal, the offending line will be ignored. When SaveSidecarFile() is next called, the offending line will be removed.

Syntax

```
void SidecarMetadataParseWarning (IBlackmagicRawClip* clip,
                                  string fileName,
                                  uint32_t lineNumber,
                                  string info)
```

Parameters

Name	Direction	Description
clip	in	Clip which was parsing the .sidecar file
fileName	in	Filename of the .sidecar file
lineNumber	in	Line number where the parse error occured
info	in	any additional information to the parse error

IBlackmagicRawCallback::SidecarMetadataParseError method

Called when a parse error occured when reading a related .sidecar file.

Note: If a parse error occurs, the entire file is ignored. When SaveSidecarFile() file is next called, the entire file will be replaced.

Syntax

```
void SidecarMetadataParseError (IBlackmagicRawClip* clip,
                                string fileName,
                                uint32_t lineNumber,
                                string info)
```

Parameters

Name	Direction	Description
clip	in	Clip which was parsing the .sidecar file
fileName	in	Filename of the .sidecar file
lineNumber	in	Line number where the parse error occured
info	in	any additional information to the parse error

IBlackmagicRawCallback::PreparePipelineComplete method

Called when preparation of the pipeline has completed

Syntax

```
void PreparePipelineComplete (void* userData, HRESULT result)
```

Parameters

Name	Direction	Description
userData	in	Userdata specified to PreparePipeline
result	in	Result of the pipeline preparation

IBlackmagicRawClipAudio Interface

Interface for accessing a clips audio.

Related Interfaces

Interface	Interface ID
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions	
Method	**Description**
`GetAudioFormat`	Get format the audio was recorded in
`GetAudioBitDepth`	Get the audio bit depth
`GetAudioChannelCount`	Get the audio channel count
`GetAudioSampleRate`	Get the audio sample rate
`GetAudioSampleCount`	Get the audio sample count per channel
`GetAudioSamples`	Get audio samples from the clip

IBlackmagicRawClipAudio::GetAudioFormat method

Get format the audio was recorded in

Syntax

```
HRESULT GetAudioFormat (BlackmagicRawAudioFormat* format)
```

Parameters

Name	Direction	Description
format	out	Returned audio format

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when format is NULL.

IBlackmagicRawClipAudio::GetAudioBitDepth method

Get the audio bit depth

Syntax

```
HRESULT GetAudioBitDepth (uint32_t* bitDepth)
```

Parameters

Name	Direction	Description
bitDepth	out	Returned audio bit depth

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when bitDepth is NULL. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipAudio::GetAudioChannelCount method

Get the audio channel count

Syntax

```
HRESULT GetAudioChannelCount (uint32_t* channelCount)
```

Parameters

Name	Direction	Description
channelCount	out	Returned audio channel count

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when channelCount is NULL. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipAudio::GetAudioSampleRate method

Get the audio sample rate

Syntax

```
HRESULT GetAudioSampleRate (uint32_t* sampleRate)
```

Parameters

Name	Direction	Description
sampleRate	out	Returned audio sample rate

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when sampleRate is NULL, E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipAudio::GetAudioSampleCount method

Get the audio sample count per channel

Syntax

```
HRESULT GetAudioSampleCount (uint64_t* sampleCount)
```

Parameters

Name	Direction	Description
sampleCount	out	Returned audio sample count per channel

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when sampleCount is NULL, E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipAudio::GetAudioSamples method

Get audio samples from the clip

Syntax

```
HRESULT GetAudioSamples (int64_t sampleFrameIndex,
                         void* buffer,
                         uint32_t bufferSizeBytes,
                         uint32_t maxSampleCount,
                         uint32_t* samplesRead,
                         uint32_t* bytesRead)
```

Parameters

Name	Direction	Description
sampleFrameIndex	in	Sample frame index to start reading from
buffer	in	Buffer to write the sample data in to
bufferSizeBytes	in	Size of the provided buffer in bytes
maxSampleCount	in	Max sample count to get with this query
samplesRead	out	Returned read sample count
bytesRead	out	Returned read byte count

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when bufferOut is NULL, E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipAccelerometerMotion Interface

Interface for accessing a clip's accelerometer motion data.

Related Interfaces

Interface	Interface ID
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions	
Method	**Description**
`GetSampleRate`	Get the motion sample rate
`GetSampleCount`	Get the motion sample count
`GetSampleSize`	Get the size (in floats) of each motion sample
`GetSampleRange`	Get motion samples for the specified range

IBlackmagicRawClipAccelerometerMotion::GetSampleRate method

Get the motion sample rate

Syntax

```
HRESULT GetSampleRate(float* sampleRate)
```

Parameters

Name	Direction	Description
sampleRate	out	Returned motion sample rate (samples per second)

IBlackmagicRawClipAccelerometerMotion::GetSampleCount method

Get the motion sample count

Syntax

```
HRESULT GetSampleCount(uint32_t* sampleCount)
```

Parameters

Name	Direction	Description
sampleCount	out	Returned motion sample count

IBlackmagicRawClipAccelerometerMotion::GetSampleSize method

Get the size (in floats) of each motion sample

Syntax

```
HRESULT GetSampleSize(uint32_t* sampleSize)
```

Parameters

Name	Direction	Description
sampleSize	out	Returned motion sample size (count of floats)

IBlackmagicRawClipAccelerometerMotion::GetSampleRange method

Get motion samples for the specified range

Syntax

```
HRESULT GetSampleRange(uint64_t sampleStartIndex,
          uint32_t sampleCount,
          float* samples,
          uint32_t* sampleCountOut)
```

Parameters

Name	Direction	Description
sampleStartIndex	in	Requested sample start index
sampleCount	in	Requested sample count
samples	out	Filled motion samples
sampleCountOut	out	Returned sample count

IBlackmagicRawClipGyroscopeMotion Interface

Interface for accessing a clip's gyroscope motion data.

Related Interfaces

Interface	Interface ID
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions	
Method	**Description**
`GetSampleRate`	Get the motion sample rate
`GetSampleCount`	Get the motion sample count
`GetSampleSize`	Get the size (in floats) of each motion sample
`GetSampleRange`	Get motion samples for the specified range

IBlackmagicRawClipGyroscopeMotion::GetSampleRate method

Get the motion sample rate

Syntax

```
HRESULT GetSampleRate(float* sampleRate)
```

Parameters

Name	Direction	Description
sampleRate	out	Returned motion sample rate (samples per second)

IBlackmagicRawClipGyroscopeMotion::GetSampleCount method

Get the motion sample count

Syntax

```
HRESULT GetSampleCount(uint32_t* sampleCount)
```

Parameters

Name	Direction	Description
sampleCount	out	Returned motion sample count

IBlackmagicRawClipGyroscopeMotion::GetSampleSize method

Get the size (in floats) of each motion sample

Syntax

```
HRESULT GetSampleSize(uint32_t* sampleSize)
```

Parameters

Name	Direction	Description
sampleSize	out	Returned motion sample size (count of floats)

IBlackmagicRawClipGyroscopeMotion::GetSampleRange method

Get motion samples for the specified range

Syntax

```
HRESULT GetSampleRange(uint64_t sampleStartIndex,
                       uint32_t sampleCount,
                       float* samples,
                       uint32_t* sampleCountOut)
```

Parameters

Name	Direction	Description
sampleStartIndex	in	Requested sample start index
sampleCount	in	Requested sample count
samples	out	Filled motion samples
sampleCountOut	out	Returned sample count

IBlackmagicRawFrame Interface

A frame that has been read but not yet processed. This is returned in the ReadComplete() callback. From here the user should prepare the frame for processing, and call DecodeAndProcessFrame(). QueryInterface can return: 1. This frames FrameProcessingAttributes, modify this to change processing attributes of this frame in the clip. 2. FrameEx

Related Interfaces

Interface	Interface ID
IBlackmagicRawMetadataIterator	IID_IBlackmagicRawMetadataIterator
IBlackmagicRawFrameProcessingAttributes	IID_IBlackmagicRawFrameProcessingAttributes
IBlackmagicRawJob	IID_IBlackmagicRawJob
BlackmagicRawFrameEx	IID_IBlackmagicRawFrameEx
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions

Method	Description
`GetFrameIndex`	Get the frameIndex
`GetTimecode`	Get a formatted timecode for this frame
`GetMetadataIterator`	Create a medatadata iterator to iterate through the metadata in this frame
`GetMetadata`	Query a single frame metadata value defined by key
`SetMetadata`	Set metadata to this frame, this data is not saved to disk until IBlackmagicRawClip::SaveSidecar() is called.
`CloneFrameProcessingAttributes`	Clone this frame's FrameProcessingAttributes into another copy. From here the returned FrameProcessingAttributes can be modified, and then provided to DecodeAndProcess() allowing the user to decode the frame with different processing attributes than specified in the clip. This is useful when the user wishes to preview different processing attributes.
`SetResolutionScale`	Set the resolution scale we want to decode this image to. This can be used to enhance turn-around time when working on the project
`GetResolutionScale`	Get the resolution scale set to the frame
`SetResourceFormat`	Set the desired resource format that we want to processing this frame in to
`GetResourceFormat`	Get the resource format this frame will be processed in to
`GetSensorRate`	Get the sensor rate with which this frame was recorded
`CreateJobDecodeAndProcessFrame`	Create a job that will decode and process our image. When completed we will receive a ProcessComplete() callback

IBlackmagicRawFrame::GetFrameIndex method

Get the frameIndex

Syntax

```
HRESULT GetFrameIndex (uint64_t* frameIndex)
```

Parameters

Name	Direction	Description
frameIndex	out	Returned frame index

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameIndex is NULL.

IBlackmagicRawFrame::GetTimecode method

Get a formatted timecode for this frame

Syntax

```
HRESULT GetTimecode (string* timecode)
```

Parameters

Name	Direction	Description
timecode	out	Returned timecode for this frame

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when timecode is NULL. E_UNEXPECTED is returned if an unexpected error occurs.

IBlackmagicRawFrame::GetMetadataIterator method

Create a medatadata iterator to iterate through the metadata in this frame

Syntax

```
HRESULT GetMetadataIterator (IBlackmagicRawMetadataIterator** iterator)
```

Parameters

Name	Direction	Description
iterator	out	Returned metadata object

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when iterator is NULL. E_FAIL can occur if the iterator failed to create.

IBlackmagicRawFrame::GetMetadata method

Query a single frame metadata value defined by key

Syntax

```
HRESULT GetMetadata (string key,
                     Variant* value).
```

Parameters

Name	Direction	Description
key	in	Key of the frame metadata entry we are looking for
value	out	Returned value of frame metadata entry at the provided key

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when key is invalid. E_POINTER is returned when value is NULL.

IBlackmagicRawFrame::SetMetadata method

Set metadata to this frame, this data is not saved to disk until IBlackmagicRawClip::SaveSidecar() is called.

Syntax

```
HRESULT SetMetadata (string key,
                     Variant* value)
```

Parameters

Name	Direction	Description
key	in	Key of the frame metadata entry we want to set. Note: to clear metadata from the sidecar and restore what was originally in the movie, set value to NULL.
value	in	Value we want to set to the frame metadata entry

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when key is invalid or value is of incorrect type. E_FAIL is returned if the metadata failed to write.

IBlackmagicRawFrame::CloneFrameProcessingAttributes method

Clone this frame's FrameProcessingAttributes into another copy. From here the returned FrameProcessingAttributes can be modified, and then provided to DecodeAndProcess() allowing the user to decode the frame with different processing attributes than specified in the clip. This is useful when the user wishes to preview different processing attributes.

Syntax

```
HRESULT CloneFrameProcessingAttributes (IBlackmagicRawFrameProcessingAttributes**
                                        frameProcessingAttributes)
```

Parameters

Name	Direction	Description
frameProcessingAttributes	out	Returned created FrameProcessingAttributes object

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameProcessingAttributes is NULL. E_FAIL can occur if the object failed to create.

IBlackmagicRawFrame::SetResolutionScale method

Set the resolution scale we want to decode this image to. This can be used to enchance turn-around time when working on the project

Syntax

```
HRESULT SetResolutionScale (BlackmagicRawResolutionScale resolutionScale)
```

Parameters

Name	Direction	Description
resolutionScale	in	Desired resolution scale

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when gamut is invalid.

IBlackmagicRawFrame::GetResolutionScale method

Get the resolution scale set to the frame

Syntax

```
HRESULT GetResolutionScale (BlackmagicRawResolutionScale* resolutionScale)
```

Parameters

Name	Direction	Description
resolutionScale	out	Returned resolution scale

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resolutionScale is NULL.

IBlackmagicRawFrame::SetResourceFormat method

Set the desired resource format that we want to processing this frame in to

Syntax

```
HRESULT SetResourceFormat (BlackmagicRawResourceFormat resourceFormat)
```

Parameters

Name	Direction	Description
resourceFormat	in	The desired resource format, see BlackmagicRawResourceFormat

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when **resourceFormat** is invalid.

IBlackmagicRawFrame::GetResourceFormat method

Get the resource format this frame will be processed in to

Syntax

```
HRESULT GetResourceFormat (BlackmagicRawResourceFormat* resourceFormat)
```

Parameters

Name	Direction	Description
resourceFormat	out	Returned resource format, see BlackmagicRawResourceFormat

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **resourceFormat** is NULL.

IBlackmagicRawFrame::GetSensorRate method

Get the sensor rate with which this frame was recorded

Syntax

```
HRESULT GetSensorRate(float* sensorRate)
```

Parameters

Name	Direction	Description
sensorRate	out	Returned sensor rate, in frames per second

IBlackmagicRawFrame::CreateJobDecodeAndProcessFrame method

Create a job that will decode and process our image. When completed we will receive a **ProcessComplete()** callback

Syntax

```
HRESULT CreateJobDecodeAndProcessFrame (IBlackmagicRawClipProcessingAttributes*
                                        clipProcessingAttributes,
                                        IBlackmagicRawFrameProcessingAttributes*
                                        frameProcessingAttributes,
                                        IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
clipProcessingAttributes	in	This allows the user to provide custom clip processing attributes which are not set to the clip. This allows the user to preview how the image would look with different settings before applying them to the clip
frameProcessingAttributes	in	This allows the user to provide custom frame processing attributes which are not set to the frame. This allows the user to preview how the image would look with different settings before applying them to the frame
job	out	Created job object used to track the job. Note: Be sure to call Submit() on the job when ready

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the decoder failed to start or the job failed to create.

IBlackmagicRawFrameEx Interface

Query additional information for the frame. This information is useful when decoding via the manual decoders.

Related Interfaces

Interface	Interface ID
IBlackmagicRawFrame	IID_IBlackmagicRawFrame

Public Member Functions	
Method	**Description**
`GetBitStreamSizeBytes`	Get the frames bistream size in bytes we've read off disk.
`GetProcessedImageResolution`	Query what the resolution of the processed image will be given the input resolution and the ResolutionScale applied

IBlackmagicRawFrameEx::GetBitStreamSizeBytes method

Get the frames bistream size in bytes we've read off disk.

Syntax

```
HRESULT GetBitStreamSizeBytes (uint32_t* bitStreamSizeBytes)
```

Parameters

Name	Direction	Description
bitStreamSizeBytes	out	Returned bitstream size in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when bitStreamSizeBytes is NULL.

IBlackmagicRawFrameEx::GetProcessedImageResolution method

Query what the resolution of the processed image will be given the input resolution and the ResolutionScale applied

Syntax

```
HRESULT GetProcessedImageResolution (uint32_t* width,
                                     uint32_t* height)
```

Parameters

Name	Direction	Description
width	out	The resultant calculated width of the processed image
height	out	The resultant calculated height of the processed image

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when width or height is NULL.

IBlackmagicRawManualDecoderFlow1 Interface

Manual decoders give you more control over which buffers are used and how things are queued. IBlackmagicRawManualDecoderFlow1 is a pure-CPU solution.

Note: these decoders are optional and targetted at advanced users

Related Interfaces

Interface	Interface ID
IBlackmagicRaw	IID_IBlackmagicRaw

Public Member Functions	
Method	**Description**
`PopulateFrameStateBuffer`	The manual decoders work with data blobs rather than API objects. This allows the user to transfer the data blob to another codec instance or potentially another computer for processing. This function converts the internal state of IBlackmagicRawFrame to frame state buffer, which is used to perform the decode

Public Member Functions	
Method	**Description**
`GetFrameStateSizeBytes`	Query the same of the FrameState buffer in bytes
`GetDecodedSizeBytes`	Query the size of the decoded buffer
`GetProcessedSizeBytes`	Query the size of the processed buffer
`GetPost3DLUTSizeBytes`	Query the size of the post 3D LUT buffer
`CreateJobDecode`	Create a job to decode a frame. After this decode is complete the decoded buffer will need to be processed to get final result. This decode completion will be notified via the OnDecodeComplete() callback
`CreateJobProcess`	Create a job to process a frame. After this process is complete a final processed image will be provided via a OnProcessComplete() callback

IBlackmagicRawManualDecoderFlow1::PopulateFrameStateBuffer method

The manual decoders work with data blobs rather than API objects. This allows the user to transfer the data blob to another codec instance or potentially another computer for processing. This function converts the internal state of IBlackmagicRawFrame to frame state buffer, which is used to perform the decode

Syntax

```
HRESULT PopulateFrameStateBuffer (IBlackmagicRawFrame* frame,
                                  IBlackmagicRawClipProcessingAttributes*
                                  clipProcessingAttributes,
                                  IBlackmagicRawFrameProcessingAttributes*
                                  frameProcessingAttributes,
                                  void* frameState,
                                  uint32_t frameStateSizeBytes)
```

Parameters

Name	Direction	Description
frame	in	Frame to read when creating a frame state
clipProcessingAttributes	in	optionally provide custom clip processing attributes to use, rather than values inside clip
frameProcessingAttributes	in	optionally provide custom frame processing attributes to use, rather than using values inside frame
frameState	out	output buffer location to store framebuffer information
frameStateSizeBytes	in	size (in bytes) of output framebuffer location

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameState is NULL. E_INVALIDARG is returned when frame is NULL or frameStateBufferSizeBytes is too small.

IBlackmagicRawManualDecoderFlow1::GetFrameStateSizeBytes method

Query the same of the FrameState buffer in bytes

Syntax

```
HRESULT GetFrameStateSizeBytes (uint32_t* frameStateSizeBytes)
```

Parameters

Name	Direction	Description
frameStateSizeBytes	out	Returns the size in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameStateSizeBytes is NULL.

IBlackmagicRawManualDecoderFlow1::GetDecodedSizeBytes method

Query the size of the decoded buffer

Syntax

```
HRESULT GetDecodedSizeBytes (void* frameStateBufferCPU,
                             uint32_t* decodedSizeBytes)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
decodedSizeBytes	out	Returns size of decoded frame in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **decodedSizeBytes** is NULL. E_INVALIDARG is returned when **frameStateBufferCPU** is invalid

IBlackmagicRawManualDecoderFlow1::GetProcessedSizeBytes method

Query the size of the processed buffer

Syntax

```
HRESULT GetProcessedSizeBytes (void* frameStateBufferCPU,
                               uint32_t* processedSizeBytes)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
processedSizeBytes	out	Returns size of processed frame in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **processedSizeBytes** is NULL. E_INVALIDARG is returned when **frameStateBufferCPU** is invalid

IBlackmagicRawManualDecoderFlow1::GetPost3DLUTSizeBytes method

Query the size of the post 3D LUT buffer

Syntax

```
HRESULT GetPost3DLUTSizeBytes (void* frameStateBufferCPU,
                               uint32_t* post3DLUTSizeBytes)
```

Parameters

Name	Direction	Description
StateBufferCPU	in	Previously prepared frame state buffer
post3DLUTSizeBytes	out	Returns size of post 3D LUT buffer in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **processedSizeBytes** is NULL. E_INVALIDARG is returned when **frameStateBufferCPU** is invalid

IBlackmagicRawManualDecoderFlow1::CreateJobDecode method

Create a job to decode a frame. After this decode is complete the decoded buffer will need to be processed to get final result. This decode completion will be notified via the **OnDecodeComplete()** callback

Syntax

```
HRESULT CreateJobDecode (void* frameStateBufferCPU, void* bitStreamBufferCPU,
                         void* decodedBufferCPU, IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
bitStreamBufferCPU	in	Previously read bitstream buffer, see BlackmagicRawClipEx::CreateJobReadFrame()
decodedBufferCPU	in	Buffer to store decoded frame in
job	out	Job created to perform the decode. Note: Remember to call job->Submit() to submit the job to the decoder!

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if frameStateBufferCPU, bitStreamBufferCPU or decodedBufferCPU is invalid. E_INVALIDARG can also be returned if SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the decoder failed to start or the job failed to create.

IBlackmagicRawManualDecoderFlow1::CreateJobProcess method

Create a job to process a frame. After this process is complete a final processed image will be provided via a **OnProcessComplete()** callback

Syntax

```
HRESULT CreateJobProcess (void* frameStateBufferCPU, void* decodedBufferCPU,
                          void* processedBufferCPU, IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
decodedBufferCPU	in	Previously decoded buffer to read from
processedBufferCPU	in	Buffer to store processed image in
post3DLUTBufferCPU	in	Post3D LUT buffer to apply, should be non-null when frameState requires it
job	out	Job created to perform the process. Note: Remember to call job->Submit() to submit the job to the decoder

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if frameStateBufferCPU, decodedBufferCPU or processedBufferCPU is invalid. E_INVALIDARG can also be returned if SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the decoder failed to start or the job failed to create.

IBlackmagicRawManualDecoderFlow2 Interface

Manual decoders give you more control over which buffers are used and how things are queued. **IBlackmagicRawManualDecoderFlow2** is a hybrid CPU/GPU solution. This will likely be faster than Flow1, however it will depend on the GPU in the users system.

Note: These decoders are optional and targetted at advanced users

Related Interfaces

Interface	Interface ID
IBlackmagicRaw	IID_IBlackmagicRaw

Public Member Functions	
Method	**Description**
`PopulateFrameStateBuffer`	The manual decoders work with data blobs rather than API objects. This allows the user to transfer the data blob to another codec instance or potentially another computer for processing. This function converts the internal state of IBlackmagicRawFrame to frame state buffer, which is used to perform the decode
`GetFrameStateSizeBytes`	Query the same of the FrameState buffer in bytes
`GetDecodedSizeBytes`	Query the size of the decoded buffer
`GetWorkingSizeBytes`	Query the size of the working buffer

Public Member Functions	
Method	**Description**
`GetProcessedSizeBytes`	Query the size of the processed buffer
`GetPost3DLUTSizeBytes`	Query the size of the post 3D LUT buffer
`CreateJobDecode`	Create a job to decode a frame. This is performed on CPU. After this decode is complete the decoded buffer will need to be processed to get final result. This decode completion will be notified via the OnDecodeComplete() callback
`CreateJobProcess`	Create a job to process a frame. This is performed on the specified GPU. After this process is complete a final processed image will be provided via a OnProcessComplete() callback

IBlackmagicRawManualDecoderFlow2::PopulateFrameStateBuffer method

The manual decoders work with data blobs rather than API objects. This allows the user to transfer the data blob to another codec instance or potentially another computer for processing. This function converts the internal state of **IBlackmagicRawFrame** to frame state buffer, which is used to perform the decode

Syntax

```
HRESULT PopulateFrameStateBuffer (IBlackmagicRawFrame* frame,
                                  IBlackmagicRawClipProcessingAttributes*
                                  clipProcessingAttributes,
                                  IBlackmagicRawFrameProcessingAttributes*
                                  frameProcessingAttributes,
                                  void* frameState,
                                  uint32_t frameStateSizeBytes)
```

Parameters

Name	Direction	Description
frame	in	Frame to read when creating a frame state
clipProcessingAttributes	in	optionally provide custom clip processing attributes to use, rather than values inside clip
frameProcessingAttributes	in	optionally provide custom frame processing attributes to use, rather than using values inside frame
frameState	out	output buffer location to store framebuffer information
frameStateSizeBytes	in	size (in bytes) of output framebuffer location

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameState is NULL. E_INVALIDARG is returned when frame is NULL or **frameStateBufferSizeBytes** is too small.

IBlackmagicRawManualDecoderFlow2::GetFrameStateSizeBytes method

Query the same of the FrameState buffer in bytes

Syntax

```
HRESULT GetFrameStateSizeBytes (uint32_t* frameStateSizeBytes)
```

Parameters

Name	Direction	Description
frameStateSizeBytes	out	Returns the size in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **frameStateSizeBytes** is NULL.

IBlackmagicRawManualDecoderFlow2::GetDecodedSizeBytes method

Query the size of the decoded buffer

Syntax

```
HRESULT GetDecodedSizeBytes (void* frameStateBufferCPU, uint32_t* decodedSizeBytes)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
decodedSizeBytes	out	Returns size of decoded frame in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when decodedSizeBytes is NULL. E_INVALIDARG is returned when frameStateBufferCPU is invalid

IBlackmagicRawManualDecoderFlow2::GetWorkingSizeBytes method

Query the size of the working buffer

Syntax

```
HRESULT GetWorkingSizeBytes (void* frameStateBufferCPU, uint32_t* workingSizeBytes)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
workingSizeBytes	out	Returns size of working buffer in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when workingSizeBytes is NULL. E_INVALIDARG is returned when frameStateBufferCPU is invalid

IBlackmagicRawManualDecoderFlow2::GetProcessedSizeBytes method

Query the size of the processed buffer

Syntax

```
HRESULT GetProcessedSizeBytes (void* frameStateBufferCPU,
                               uint32_t* processedSizeBytes)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
processedSizeBytes	out	Returns size of the processed buffer in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **processedSizeBytes** is NULL. E_INVALIDARG is returned when frameStateBufferCPU is invalid

IBlackmagicRawManualDecoderFlow2::GetPost3DLUTSizeBytes method

Query the size of the post 3D LUT buffer

Syntax

```
HRESULT GetPost3DLUTSizeBytes (void* frameStateBufferCPU,uint32_t* post3DLUTSizeBytes)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Previously prepared frame state buffer
post3DLUTSizeBytes	out	Returns size of post 3D LUT buffer in bytes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when **processedSizeBytes** is NULL. E_INVALIDARG is returned when frameStateBufferCPU is invalid

IBlackmagicRawManualDecoderFlow2::CreateJobDecode method

Create a job to decode a frame. This is performed on CPU. After this decode is complete the decoded buffer will need to be processed to get final result. This decode completion will be notified via the **OnDecodeComplete()** callback

Syntax

```
HRESULT CreateJobDecode (void* frameStateBufferCPU,
                         void* bitStreamBufferCPU,
                         void* decodedBufferCPU,
                         IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
frameStateBufferCPU	in	Query the size of the processed buffer. Note: this is a CPU resource (and thus stored in CPU memory)
bitStreamBufferCPU	in	Previously read bitream buffer, see **BlackmagicRawClipEx::CreateJobReadFrame().** Note: this is a CPU resource (and thus stored in CPU memory)
decodedBufferCPU	in	CPU resource where we the decoded buffer will be written to. Note: this is a CPU resource (and thus stored in CPU memory)
job	out	Job created to perform the decode. Note: Remember to call job->Submit() to submit the job to the decoder!

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if **frameStateBufferCPU**, **bitStreamBufferCPU** or **decodedBufferCPU** is invalid. E_INVALIDARG can also be returned if **SetCallback()** hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the decoder failed to start or the job failed to create.

IBlackmagicRawManualDecoderFlow2::CreateJobProcess method

Create a job to process a frame. This is performed on the specified GPU. After this process is complete a final processed image will be provided via a **OnProcessComplete()** callback

Syntax

```
HRESULT CreateJobProcess (void* context,
                          void* commandQueue,
                          void* frameStateBufferCPU,
                          void* decodedBufferGPU,
                          void* workingBufferGPU,
                          void* processedBufferGPU,
                          IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
context	in	Context to perform the process on. This will be API dependant, see BlackmagicRawPipeline for details
commandQueue	in	Command queue to perform the process on. This will be API dependant, see **BlackmagicRawPipeline** for details
frameStateBufferCPU	in	Previously prepared frame state buffer. Note: this is a CPU resource (and thus stored in CPU memory)
decodedBufferGPU	in	GPU resource where the decoded buffer has been decoded in to. Note: this is a GPU resource, and its type will differ depending on API, see **BlackmagicRawResourceType**. Note: The users responsibility to transfer the decoded buffer from CPU to GPU before calling this function.
workingBufferGPU	in	An additional GPU resource uses as working memory
processedBufferGPU	in	Resource to store the processed buffer in to. Note: this is a GPU resource, and thus it's type will be API dependant, see **BlackmagicRawPipeline** for details
post3DLUTBufferGPU	in	Post3D LUT buffer to apply, should be non-null when frameState requires it
job	out	Job created to perform the process. Note: Remember to call job->Submit() to submit the job to the decoder

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if context, commandQueue, frameStateBufferCPU, decodedBufferGPU, workingBufferGPU or processedBufferGPU is invalid. E_INVALIDARG can also be returned if SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the decoder failed to start or the job failed to create.

IBlackmagicRawClip Interface

Clip object, created by calling **IBlackmagicRaw::OpenClip**()

Related Interfaces

Interface	Interface ID
IBlackmagicRaw	IID_IBlackmagicRaw
IBlackmagicRawClipAccelerometerMotion	IID_IBlackmagicRawClipAccelerometerMotion
IBlackmagicRawClipGyroscopeMotion	IID_IBlackmagicRawClipGyroscopeMotion
IBlackmagicRawFrame	IID_IBlackmagicRawFrame
IBlackmagicRawMetadataIterator	IID_IBlackmagicRawMetadataIterator
IBlackmagicRawClipProcessingAttributes	IID_IBlackmagicRawClipProcessingAttributes
IBlackmagicRawJob	IID_IBlackmagicRawJob
IBlackmagicRawClipEx	IID_IBlackmagicRawClipEx
IBlackmagicRawClipAudio	IID_IBlackmagicRawClipAudio
IBlackmagicRawClipOrientationMotion	IID_IBlackmagicRawClipOrientationMotion

Public Member Functions	
Method	**Description**
`GetWidth`	Get the width of the clip
`GetHeight`	Get the height of the clip
`GetFrameRate`	Get the frame rate of the clip
`GetFrameCount`	Get the frame count in the clip
`GetTimecodeForFrame`	Get the timecode for the specified frame
`GetMetadataIterator`	Create a medatadata iterator to iterate through the metadata in this clip
`GetMetadata`	Query a single clip metadata value defined by key
`SetMetadata`	Set metadata to this clip, this data is not saved to disk until IBlackmagicRawClip::SaveSidecar() is called
`GetCameraType`	Get the camera type on which this clip was recorded
`CloneClipProcessingAttributes`	Clone this clip's ClipProcessingAttributes into another copy. From here the returned ClipProcessingAttributes can be modified, and then provided to DecodeAndProcess() allowing the user to decode the frame with different processing attributes than specified in the clip. This is useful when the user wishes to preview different processing attributes.
`GetMulticardFileCount`	Queries how many cards this movie was originally recorded on to
`IsMulticardFilePresent`	Queries if a particular card file from the original recording are present. If files are missing the movie will still play back, just at a lower framerate

Public Member Functions	
Method	**Description**
`GetSidecarFileAttached`	Returns if a relevant .sidecar file was present on disk
`SaveSidecarFile`	This will save all set metadata and processing attributes to the .sidecar file on disk. From here the clip can be safely closed and data will be preserved
`ReloadSidecarFile`	Reload the .sidecar file, this will replace all previously non-saved metadata and processing attributes with the contents of the .sidecar file
`CreateJobReadFrame`	Create a job that will read the frames bitstream into memory. When completed we will receive a ReadComplete() callback
`CreateJobTrim`	A trim will export part of the clip with the .sidecar file baked in to a new .braw file. This is an asynchronous job and can take some time depending on the length of the trim

IBlackmagicRawClip::GetWidth method

Get the width of the clip

Syntax

```
HRESULT GetWidth (uint32_t* width)
```

Parameters

Name	Direction	Description
width	out	Returns the width of the clip, in pixels

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when width is NULL.

IBlackmagicRawClip::GetHeight method

Get the height of the clip

Syntax

```
HRESULT GetHeight (uint32_t* height)
```

Parameters

Name	Direction	Description
height	out	Returns the height of the clip, in pixels

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when height is NULL.

IBlackmagicRawClip::GetFrameRate method

Get the frame rate of the clip

Syntax

```
HRESULT GetFrameRate (float* frameRate)
```

Parameters

Name	Direction	Description
frameRate	out	Returns the frame rate of the clip, in frames per second

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameRate is NULL. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClip::GetFrameCount method

Get the frame count in the clip

Syntax

```
HRESULT GetFrameCount (uint64_t* frameCount)
```

Parameters

Name	Direction	Description
frameCount	out	Returns the number of frames in the clip

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when frameCount is NULL.

IBlackmagicRawClip::GetTimecodeForFrame method

Get the timecode for the specified frame

Syntax

```
HRESULT GetTimecodeForFrame (uint64_t frameIndex, string* timecode)
```

Parameters

Name	Direction	Description
frameIndex	in	Index of the frame we are querying
timecode	out	Returns a formatted timecode for the specified frame

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when frameIndex is out of range. E_POINTER is returned when timecode is NULL.

IBlackmagicRawClip::GetMetadataIterator method

Create a medatadata iterator to iterate through the metadata in this clip

Syntax

```
HRESULT GetMetadataIterator (IBlackmagicRawMetadataIterator** iterator)
```

Parameters

Name	Direction	Description
iterator	out	Returned metadata object

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when iterator is NULL. E_FAIL can occur if the iterator failed to create.

IBlackmagicRawClip::GetMetadata method

Query a single clip metadata value defined by key

Syntax

```
HRESULT GetMetadata (string key,
                     Variant* value)
```

Parameters

Name	Direction	Description
key	in	Key of the clip metadata entry we are looking for
value	out	Returned value of clip metadata entry at the provided key

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when key is invalid. E_POINTER is returned when value is NULL.

IBlackmagicRawClip::SetMetadata method

Set metadata to this clip, this data is not saved to disk until IBlackmagicRawClip::SaveSidecar() is called

Syntax

```
HRESULT SetMetadata (string key,
                     Variant* value)
```

Parameters

Name	Direction	Description
key	in	Key of the clip metadata entry we want to set. Note: to clear metadata from the sidecar and restore what was originally in the movie, set value to NULL.
value	in	Value we want to set to the clip metadata entry

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when key is invalid or value is of incorrect type. E_FAIL is returned if the metadata failed to write.

IBlackmagicRawClip::GetCameraType method

Get the camera type on which this clip was recorded

Syntax

```
HRESULT GetCameraType (string* cameraType)
```

Parameters

Name	Direction	Description
cameraType	out	Returned camera type. This string can be used for display purposes

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when cameraType is NULL.

IBlackmagicRawClip::CloneClipProcessingAttributes method

Clone this clip's ClipProcessingAttributes into another copy. From here the returned ClipProcessingAttributes can be modified, and then provided to DecodeAndProcess() allowing the user to decode the frame with different processing attributes than specified in the clip. This is useful when the user wishes to preview different processing attributes.

Syntax

```
HRESULT CloneClipProcessingAttributes
(IBlackmagicRawClipProcessingAttributes** clipProcessingAttributes)
```

Parameters

Name	Direction	Description
clipProcessingAttributes	out	Returned created ClipProcessingAttributes object

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when clipProcessingAttributes is NULL. E_FAIL can occur if the object failed to create.

IBlackmagicRawClip::GetMulticardFileCount method

Queries how many cards this movie was originally recorded on to

Syntax

```
HRESULT GetMulticardFileCount (uint32_t* multicardFileCount)
```

Parameters

Name	Direction	Description
multicardFileCount	out	Returned multicard file count

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when multicardFileCount is NULL.

IBlackmagicRawClip::IsMulticardFilePresent method

Queries if a particular card file from the original recording are present. If files are missing the movie will still play back, just at a lower framerate

Syntax

```
HRESULT IsMulticardFilePresent (uint32_t index,
                                Boolean* isMulticardFilePresent)
```

Parameters

Name	Direction	Description
index	in	Frame index to query
isMulticardFilePresent	out	Returned boolean indicating if this file was present

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when isMulticardFilePresent is NULL.

IBlackmagicRawClip::GetSidecarFileAttached method

Returns if a relevant .sidecar file was present on disk

Syntax

```
HRESULT GetSidecarFileAttached (Boolean* isSidecarFileAttached)
```

Parameters

Name	Direction	Description
isSidecarFileAttached	out	Returned boolean indicating if the .sidecar file was present on disk

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when isSidecarFileAttached is NULL.

IBlackmagicRawClip::SaveSidecarFile method

This will save all set metadata and processing attributes to the .sidecar file on disk. From here the clip can be safely closed and data will be preserved

Syntax

```
HRESULT SaveSidecarFile()
```

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned if the save operation failed.

IBlackmagicRawClip::ReloadSidecarFile method

Reload the .sidecar file, this will replace all previously non-saved metadata and processing attributes with the contents of the .sidecar file

Syntax

```
HRESULT ReloadSidecarFile()
```

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned if the load operation failed.

IBlackmagicRawClip::CreateJobReadFrame method

Create a job that will read the frames bitstream into memory. When completed we will receive a **ReadComplete()** callback

Syntax

```
HRESULT CreateJobReadFrame (uint64_t frameIndex, IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
frameIndex	in	The frame index to read
job	out	Created job object used to track the job. Note: Be sure to call **Submit()** on the job when ready

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if frameIndex is out of range or SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the job failed to create.

IBlackmagicRawClip::CreateJobTrim method

A trim will export part of the clip with the .sidecar file baked in to a new .braw file. This is an asynchronous job and can take some time depending on the length of the trim

Syntax

```
HRESULT CreateJobTrim (string fileName,
                       uint64_t frameIndex,
                       uint64_t frameCount,
                       IBlackmagicRawClipProcessingAttributes*
                       clipProcessingAttributes,
                       IBlackmagicRawFrameProcessingAttributes*
                       frameProcessingAttributes,
                       IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
fileName	in	Target file name where to write the trimmed movie
frameIndex	in	The frame index to start trimming at
frameCount	in	The number of frames we want to trim
clipProcessingAttributes	in	Processing attributes to be applied to the trimmed clip
frameProcessingAttributes	in	Processing attributes to be applied to each frame of the trimmed clip
job	out	Created job object used to track the job. Note: Be sure to call Submit() on the job when ready

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when job is NULL. E_INVALIDARG is returned if SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the job failed to create.

IBlackmagicRawClipEx Interface

Extended use of IBlackmagicRawClip, to pass custom bitstream

Related Interfaces

Interface	Interface ID
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions	
Method	**Description**
`GetMaxBitStreamSizeBytes`	Inspects all frames in the movie and will return the maximum bit stream size encountered.
`GetBitStreamSizeBytes`	Returns the bitsream size for the provided frame
`CreateJobReadFrame`	Create a job that will read the frames bitstream into memory. When completed we will receive a ReadComplete() callback. This extended variation allows the user to control exactly where the bistream is stored in memory.
`QueryTimecodeInfo`	Queries the timecode info for the clip. This information can be used to externally calculate valid timecodes from a frameIndex. Alternatively you can call IBlackmagicRawFrame::GetTimecode() on a frame object

IBlackmagicRawClipEx::GetMaxBitStreamSizeBytes method

Inspects all frames in the movie and will return the maximum bit stream size encountered.

Syntax

```
HRESULT GetMaxBitStreamSizeBytes (uint32_t* maxBitStreamSizeBytes)
```

Parameters

Name	Direction	Description
maxBitStreamSizeBytes	out	The maximum bit stream size in bytes, for any frame in the clip

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when maxBitStreamSizeBytes is NULL.

IBlackmagicRawClipEx::GetBitStreamSizeBytes method

Returns the bitsream size for the provided frame

Syntax

```
HRESULT GetBitStreamSizeBytes (uint64_t frameIndex,
                               uint32_t* bitStreamSizeBytes)
```

Parameters

Name	Direction	Description
frameIndex	in	The frame index to query
bitStreamSizeBytes	out	Returned maximum bitstream size found in bytes.

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when bitStreamSizeBytes is NULL. E_INVALIDARG is returned when frameIndex is invalid. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipEx::CreateJobReadFrame method

Create a job that will read the frames bitstream into memory. When completed we will receive a ReadComplete() callback. This extended variation allows the user to control exactly where the bistream is stored in memory.

Syntax

```
HRESULT CreateJobReadFrame (uint64_t frameIndex,
                            void* bitStream,
                            uint32_t bitStreamSizeBytes,
                            IBlackmagicRawJob** job)
```

Parameters

Name	Direction	Description
frameIndex	in	The frame index to read
bitStream	out	output CPU resource (i.e. memory address) where the frame's bitstream data is written to.
bitStreamSizeBytes	in	size of the bitstream buffer (in bytes) the frame data is being written to.
job	out	Created job object used to track the job. Note: Be sure to call Submit() on the job when ready

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when bitStream or job is NULL. E_INVALIDARG is returned if frameIndex is out of range or bitStreamSizeBytes is 0. E_INVALIDARG is also returned if SetCallback() hasn't been called on the related BlackmagicRaw object. E_FAIL can occur if the job failed to create.

IBlackmagicRawClipEx::QueryTimecodeInfo method

Queries the timecode info for the clip. This information can be used to externally calculate valid timecodes from a frameIndex. Alternatively you can call IBlackmagicRawFrame::GetTimecode() on a frame object

Syntax

```
HRESULT QueryTimecodeInfo (uint32_t* baseFrameIndex,
                           Boolean* isDropFrameTimecode)
```

Parameters

Name	Direction	Description
baseFrameIndex	out	Frame index (at the clips framerate) where the timecode begins.
isDropFrameTimecode	out	Returns whether this movie has a drop frame timecode or not.

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when baseFrameIndex or isDropFrameTimecode is NULL. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipResolutions Interface

Supports querying of resolutions and/or scales for processed image results

Public Member Functions	
Method	**Description**
`GetResolutionCount`	Returns the number of resolutions at which the clip may be processed
`GetResolution`	Returns a resolution at which the clip may be processed
`GetClosestResolutionForScale`	Returns a resolution which most closely matches the given scale
`GetClosestScaleForResolution`	Returns a BlackmagicRawResolutionScale which most closely matches the given resolution

IBlackmagicRawClipResolutions::GetResolutionCount method

Returns the number of resolutions at which the clip may be processed

Syntax

```
HRESULT GetResolutionCount(uint32_t* resolutionCount)
```

Parameters

Name	Direction	Description
resolutionCount	out	Returned number of resolutions at which the clip may be processed.

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned if resolutionCount is NULL. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipResolutions::GetResolution method

Returns a resolution at which the clip may be processed

Syntax

```
HRESULT GetResolution(uint32_t resolutionIndex,
        uint32_t* resolutionWidthPixels,
        uint32_t* resolutionHeightPixels)
```

Parameters

Name	Direction	Description
resolutionIndex	in	The resolution index to query
resolutionWidthPixels	out	Returned resolution width in pixels.
resolutionHeightPixels	out	Returned resolution height in pixels.

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipResolutions::GetClosestResolutionForScale method

Returns a resolution which most closesly matches the given scale

Syntax

```
HRESULT GetClosestResolutionForScale(BlackmagicRawResolutionScale resolutionScale,
        uint32_t* resolutionWidthPixels,
        uint32_t* resolutionHeightPixels)
```

Parameters

Name	Direction	Description
resolutionScale	in	Desired resolution scale
resolutionWidthPixels	out	Returned resolution width in pixels.
resolutionHeightPixels	out	Returned resolution height in pixels.

Return Values

If the method succeeds, the return value is S_OK. E_FAIL is returned if an error occured when reading the movie.

IBlackmagicRawClipResolutions::GetClosestScaleForResolution method

Returns a BlackmagicRawResolutionScale which most closely matches the given resolution

Syntax

```
HRESULT GetClosestScaleForResolution(uint32_t resolutionWidthPixels,
        uint32_t resolutionHeightPixels,
        Boolean requestUpsideDown,
        BlackmagicRawResolutionScale* resolutionScale)
```

Parameters

Name	Direction	Description
resolutionWidthPixels	in	Desired resolution width in pixels.
resolutionHeightPixels	in	Desired resolution height in pixels.
requestUpsideDown	in	Request scale to render frame upside down.
resolutionScale	out	Returned resolution scale

Return Values

If the method succeeds, the return value is S_OK. E_POINTER is returned when resolutionScale is NULL. E_FAIL is returned if an error occured when reading the movie.