Blackmagicdesign

Developer Information

Blackmagic RAW SDK

December 2018

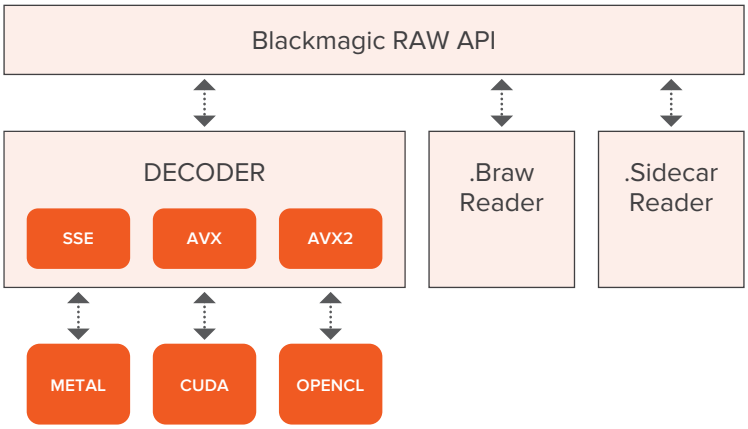
Contents

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.

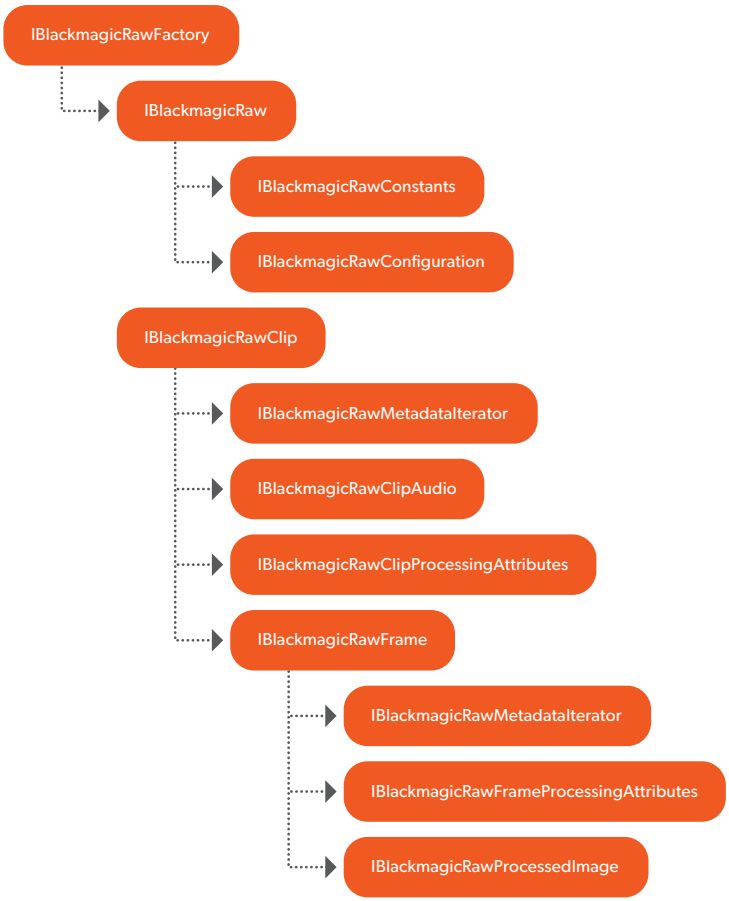
When metadata or image processing values (such as white balance) are modified via the SDK, the user can then choose to save this data to the sidecar file. Now when the movie is loaded (potentially in a different application) the sidecar file be applied and the picture will look consistent.

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.

2.0 Interface Overview

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



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 2.4 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.

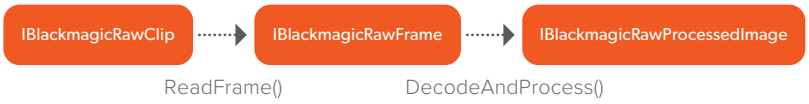
3.0 SDK Operations and flow

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.

Recommended UI Controls and Behavior

Decode Quality

Type: **Drop down selector**

Default: Full Res.

Options:

- **Full Res.**
- **1/2 Res.**
- **1/4 Res.**
- **1/8 Res.**

Color Science Version

Type: **Drop down selector**

Default: Read from metadata

Options: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeColorScienceGen

Color Space/Gamut

Type: **Drop down selector**

Default: Read from metadata

Options: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeGamut

Gamma

Type: **Drop down selector**

Default: Read from metadata.

Options: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeGamma

ISO

Type: **Drop down selector**

Default: Read from metadata.

Options: Use IBlackmagicRawConstants::GetFrameProcessingAttributeRange()
- blackmagicRawFrameProcessingAttributeISO

Exposure

Type: **Slider**

Default: 0.

Range: Use IBlackmagicRawConstants::GetFrameProcessingAttributeRange()
- blackmagicRawFrameProcessingAttributeExposure

Color Temp

Type: **Slider**

Default: 5600

Range: Use IBlackmagicRawConstants::GetFrameProcessingAttributeRange()
- blackmagicRawFrameProcessingAttributeWhiteBalanceKelvin

Tint

Type: **Slider**

Default: 10

Range: Use IBlackmagicRawConstants::GetFrameProcessingAttributeRange()
- blackmagicRawFrameProcessingAttributeWhiteBalanceTint

Highlight Recovery

Type: **Checkbox**

Default: Off

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.

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

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 IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveSaturation

Contrast

Type: **Slider**

Default: 0.5

Range: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveContrast

Midpoint

Type: **Slider**

Default: 0.41

Range: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveMidpoint

Highlight Rolloff

Type: **Slider**

Default: 1.0

Range: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveHighlights

Shadow Rolloff

Type: **Slider**

Default: 1.0

Range: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveShadows

Black Level

Type: **Slider**

Default: 1.0

Range: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveBlackLevel

White Level

Type: **Slider**

Default: 1.0

Range: Use IBlackmagicRawConstants::GetClipProcessingAttributeList()
- blackmagicRawClipProcessingAttributeToneCurveWhiteLevel

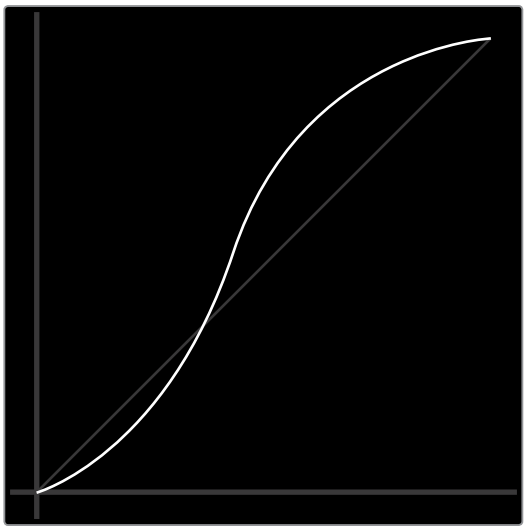
Set Video Black Level

Type: **Checkbox**

Default: Off

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:



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 signed 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

Key	Value (macOS, Linux)	Value (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
blackmagicRawResourceFormatRGBU16Planar	‘16pl'	Unsigned 16bit planar RGB
blackmagicRawResourceFormatRGBF32	‘f32s'	Floating point interleaved RGB
blackmagicRawResourceFormatRGBF32Planar	‘f32p'	Floating point planar RGB

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'	
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
blackmagicRawClipProcessingAttributeToneCurveWhiteLevel	'twit'	WhiteLevel used in Blackmagic Design Custom Gamma
blackmagicRawClipProcessingAttributeHighlightRecovery	'hlry'	Is highlight recovery enabled
blackmagicRawClipProcessingAttributeAnalogGain	'gain'	Analog Gain set at record time, cannot be changed

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

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
IBlackmagicRawConfiguration	IID_IBlackmagicRawConfiguration
IBlackmagicRawConfigurationEx	IID_IBlackmagicRawConfigurationEx
IBlackmagicRawConstants	IID_IBlackmagicRawConstants
IBlackmagicRawManualDecoderFlow1	IID_IBlackmagicRawManualDecoderFlow1
IBlackmagicRawManualDecoderFlow2	IID_IBlackmagicRawManualDecoderFlow2
IBlackmagicRawToneCurve	IID_IBlackmagicRawToneCurve

Public Member Functions

Method	Description
OpenClip	Opens a clip
SetCallback	Registers a callback with the codec object
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::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

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.

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 Clip::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 Clip::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.

IBlackmagicRawConstants Interface

If desired, the user application can cache these results

Related Interfaces

Interface	Interface ID
IBlackmagicRaw	IID_IBlackmagicRaw

Public Member Functions	
Method	**Description**
GetClipProcessingAttributeRange	Get the clip processing attribute range for the specified attribute
GetClipProcessingAttributeList	Get the clip processing attribute value list for the specified attribute
GetFrameProcessingAttributeRange	Get the frame processing attribute range for the specified attribute
GetFrameProcessingAttributeList	Get the frame processing attribute value list for the specified attribute
GetISOListForAnalogGain	The ISO selection can be constrained by the analog gain selected at record time, use this function to get a narrowed view of the available ISOs for a given cameraType and analog gain values.

IBlackmagicRawConstants:: GetClipProcessingAttributeRange method

Get the clip processing attribute range for the specified attribute

Syntax

```
HRESULT GetClipProcessingAttributeRange (string cameraType,
                                         BlackmagicRawClipProcessingAttribute
                                         attribute,
                                         Variant valueMin,
                                         Variant valueMax)
```

Parameters

Name	Direction	Description
cameraType	in	Type of camera, you can query this from Clip::GetCameraType()
attribute	in	Attribute to query
valueMin	out	Variant to store the data in
valueMax	out	Variant to store the data in

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when attribute is invalid, or the attribute does not have a range, see GetClipProcessingAttributeList(). E_POINTER is returned when valueMin or valueMax is NULL.

IBlackmagicRawConstants:: GetClipProcessingAttributeList method

Get the clip processing attribute value list for the specified attribute

Syntax

```
HRESULT GetClipProcessingAttributeList (string cameraType,
                                        BlackmagicRawClipProcessingAttribute
                                        attribute,
                                        Variant array,
                                        uint32_t* arrayElementCount)
```

Parameters

Name	Direction	Description
cameraType	in	Type of camera, you can query this from Clip::GetCameraType()
attribute	in	Attribute to query
array	out	Array the results will be written to
arrayElementCount	out	Array element count

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when attribute is invalid, or the attribute does not have a value list, see GetClipProcessingAttributeRange().

IBlackmagicRawConstants:: GetFrameProcessingAttributeRange method

Get the frame processing attribute range for the specified attribute

Syntax

```
HRESULT GetFrameProcessingAttributeRange (string cameraType,
                                          BlackmagicRawFrameProcessingAttribute
                                          attribute,
                                          Variant valueMin,
                                          Variant valueMax)
```

Parameters

Name	Direction	Description
cameraType	in	Type of camera, you can query this from Clip::GetCameraType()
attribute	in	Attribute to query
valueMin	out	Variant to store the data in
valueMax	out	Variant to store the data in

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when attribute is invalid, or the attribute does not have a range, see GetFrameProcessingAttributeList(). E_POINTER is returned when valueMin or valueMax is NULL.

IBlackmagicRawConstants:: GetFrameProcessingAttributeList method

Get the frame processing attribute value list for the specified attribute

Syntax

```
HRESULT GetFrameProcessingAttributeList (string cameraType,
                                         BlackmagicRawClipProcessingAttribute
                                         attribute,
                                         Variant array,
                                         uint32_t* arrayElementCount)
```

Parameters

Name	Direction	Description
cameraType	in	Type of camera, you can query this from Clip::GetCameraType()
attribute	in	Attribute to query
array	out	Array the results will be written to
arrayElementCount	out	Array element count

Return Values

If the method succeeds, the return value is S_OK. E_INVALIDARG is returned when attribute is invalid, or the attribute does not have a value list, see GetFrameProcessingAttributeRange().

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

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.

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

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.

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
IBlackmagicRawClip	IID_IBlackmagicRawClip

Public Member Functions

Method	Description
GetAttribute	Get the attribute
SetAttribute	Set the attribute

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.

IBlackmagicRawFrameProcessingAttributes Interface

Processing attributes which can change per frame

Related Interfaces

Interface	Interface ID
IBlackmagicRawFrame	IID_IBlackmagicRawFrame

Public Member Functions

Method	Description
GetAttribute	Get the attribute
SetAttribute	Set the attribute

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

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.

IBlackmagicRawFrameProcessingAttributes:: SetFrameAttribute method

Set the attribute

Syntax

```
HRESULT SetAttribute (BlackmagicRawFrameProcessingAttribute 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.

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 resouce, see BlackmagicRawResourceType
GetResourceFormat	Get format of resource, see BlackmagicRawResourceFormat
GetResourceSizeBytes	Get size of resouce 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 resouce, 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 resouce 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 attache

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.

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
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

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
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

Syntax

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

Parameters

Name	Direction	Description
sampleCount	out	Returned audio sample count

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.

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
IBlackmagicRawFrameEx	IID_IBlackmagicRawFrameEx
IBlackmagicRawFrameProcessingAttributes	IID_IBlackmagicRawFrameProcessingAttributes

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
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:: 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
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
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:: 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
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 size of the FrameState buffer in bytes
GetDecodedSizeBytes	Query the size of the decoded buffer
GetWorkingSizeBytes	Query the size of the working buffer
GetProcessedSizeBytes	Query the size of the processed 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::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
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
IBlackmagicRawClipEx	IID_IBlackmagicRawClipEx
IBlackmagicRawClipAudio	IID_IBlackmagicRawClipAudio
IBlackmagicRawClipProcessingAttributes	IID_IBlackmagicRawClipProcessingAttributes

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 that this clip was recorded on
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
GetSidecarFileAttached	Returns if a relevant .sidecar file was present on disk

Public Member Functions	
Method	**Description**
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 that this clip was recorded on

Syntax

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

Parameters

Name	Direction	Description
cameraType	out	Returned camera type. This string can be used for display purposes and/or query limits in IBlackmagicRawConstants:: GetBlackmagicColorScienceGenList() / IBlackmagicRawConstants::GetISOList()

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.