Blackmagicdesign

Software Developers Kit

SDK-DeckLink

June 2018

macOS™

Windows™

Linux™

Contents

Introduction

1.1 Welcome

Thanks for downloading the Blackmagic Design DeckLink Software Developers Kit.

1.2 Overview

The DeckLink SDK provides a stable, cross-platform interface to Blackmagic Design capture and playback products.

The SDK provides both low-level control of hardware and high-level interfaces to allow developers to easily perform common tasks.

The SDK consists of a set of interface descriptions & sample applications which demonstrate the use of the basic features of the hardware.

The details of the SDK are described in this document. The SDK supports Microsoft Windows, macOS and Linux platforms.

The libraries supporting the Blackmagic SDK are shipped as part of the product installers for each supported product line. Applications built against the interfaces shipped in the SDK will dynamically link against the library installed on the end-user's system.

The SDK interface is modeled on Microsoft's Component Object Model (COM). On Microsoft Windows platforms, it is provided as a native COM interface registered with the operating system. On other platforms application code is provided to allow the same COM style interface to be used.

The COM model provides a paradigm for creating flexible and extensible interfaces with minimal overhead.

You can download the Decklink SDK from the Blackmagic Design support center at:
www.blackmagicdesign.com/support

The product family is Capture and Playback.

The Blackmagic Design Developer website provides video tutorials and FAQs for developing software for Desktop Video products.

Please visit at www.blackmagicdesign.com/developer

If you're looking for detailed answers regarding technologies used by Blackmagic Design, such as codecs, core media, APIs, SDK and more, visit the Blackmagic Software Developers Forum. The forum is a helpful place for you to engage with both Blackmagic support staff and other forum members who can answer developer specific questions and provide further information. The Software Developers Forum can be found within the Blackmagic Design Forum at **forum.blackmagicdesign.com**

If you wish to ask questions outside of the software developers forum, please contact us at:
developer@blackmagicdesign.com

Section 1 - API Design

1.3 API Design

1.3.1 Supported Products

The DeckLink SDK provides programmatic access to a wide variety of Blackmagic Design products. The term “DeckLink” is used as a generic term to refer to the supported products.

Playback and Capture support is provided for devices in the DeckLink, Intensity, UltraStudio and Teranex product lines. Capture support is provided for the Cintel Scanner, Cinema Camera and Hyperdeck Studio products.

1.3.2 Supported Operating Systems

The DeckLink SDK is supported on macOS, Windows and Linux operating systems. The release notes supplied with the DeckLink packages include details of supported operating system versions.

1.3.3 3rd Party Product and Feature Support

1.3.3.1 NVIDIA GPUDirect support

NVIDIA GPUDirect is supported on Windows and Linux for x86 and x64 architectures where those platforms are also supported by NVIDIA. GPUDirect support requires the use of the DVP library supplied by NVIDIA.

See the LoopThroughWithOpenGLCompositing for a detailed example of integrating the DeckLink API and NVIDIA GPUDirect.

1.3.3.2 AMD DirectGMA support

AMD DirectGMA is supported on Windows and Linux for x86 and x64 architectures where those platforms are also supported by AMD. DirectGMA support requires the use of the GL_AMD_pinned_memory GL extension supported by compatible AMD OpenGL drivers.

See the LoopThroughWithOpenGLCompositing for a detailed example of integrating the DeckLink API and AMD DirectGMA.

1.3.4 Object Interfaces

The API provides high-level interfaces to allow capture & playback of audio and video with frame buffering and scheduling as well as low-level interfaces for controlling features available on different capture card models.

Functionality within the API is accessed via “object interfaces”. Each object in the system may inherit from and be accessed via a number of object interfaces. Typically the developer is able to interact with object interfaces and leave the underlying objects to manage themselves.

Each object interface class has a Globally Unique ID (GUID) called an “Interface ID”. On platforms with native COM support, an IID may be used to obtain a handle to an exported interface object from the OS, which is effectively an entry point to an installed API.

Each interface may have related interfaces that are accessed by providing an IID to an existing object interface (see **IUnknown::QueryInterface**). This mechanism allows new interfaces to be added to the API without breaking API or ABI compatibility.

1.3.5 Reference Counting

The API uses reference counting to manage the life cycle of object interfaces. The developer may need to add or remove references on object interfaces (see **IUnknown::AddRef** and **IUnknown::Release**) to influence their life cycle as appropriate in the application.

1.3.6 Interface Stability

The SDK provides a set of stable interfaces for accessing Blackmagic Design hardware. Whilst the published interfaces will remain stable, developers need to be aware of some issues they may encounter as new products, features and interfaces become available.

1.3.6.1 New Interfaces

Major pieces of new functionality may be added to the SDK as a whole new object interface. Already released applications will not be affected by the additional functionality. Developers making use of the new functionality should be sure to check the return of **CoCreateInstance** and/or **QueryInterface** as these interfaces will not be available on users systems which are running an older release of the Blackmagic drivers.

Developers can choose to either reduce the functionality of their application when an interface is not available, or to notify the user that they must install a later version of the Blackmagic drivers.

1.3.6.2 Updated Interfaces

As new functionality is added to the SDK, some existing interfaces may need to be modified or extended. To maintain compatibility with released software, the original interface will be deprecated but will remain available and maintain its unique identifier (IID). The replacement interface will have a new identifier and remain as similar to the original as possible.

1.3.6.3 Deprecated Interfaces

Interfaces which have been replaced with an updated version, or are no longer recommended for use are “deprecated”. Deprecated interfaces are moved out of the main interface description files into an interface description file named according to the release in which the interface was deprecated. Deprecated interfaces are also renamed with a suffix indicating the release prior to the one in which they were deprecated.

It is recommended that developers update their applications to use the most recent SDK interfaces when they release a new version of their applications. As an interim measure, developers may include the deprecated interface descriptions, and updating the names of the interfaces in their application to access the original interface functionality.

1.3.6.4 Removed Interfaces

Interfaces that have been deprecated for some time may eventually be removed in a major driver update if they become impractical to support.

1.4 Interface Reference

Every object interface subclasses the **IUnknown** interface.

1.4.1 IUnknown Interface

Each API interface is a subclass of the standard COM base class – **IUnknown**. The **IUnknown** object interface provides reference counting and the ability to look up related interfaces by interface ID. The interface ID mechanism allows interfaces to be added to the API without impacting existing applications.

Public Member Functions	
Method	**Description**
QueryInterface	Provides access to supported child interfaces of the object.
AddRef	Increments the reference count of the object.
Release	Decrements the reference count of the object. When the final reference is removed, the object is freed.

1.4.1.1 IUnknown::QueryInterface method

The **QueryInterface** method looks up a related interface of an object interface.

Syntax

```
HRESULT      QueryInterface(REFIID id, void **outputInterface);
```

Parameters

Name	Direction	Description
`id`	in	Interface ID of interface to lookup
`outpuInterface`	out	New object interface or NULL on failure

Return Values

Value	Description
E_NOINTERFACE	Interface was not found.
S_OK	Success.

1.4.1.2 IUnknown::AddRef method

The **AddRef** method increments the reference count for an object interface.

Syntax

```
ULONG        AddRef();
```

Return Values

Value	Description
Count	New reference count – for debug purposes only.

1.4.1.3 IUnknown::Release method

The **Release** method decrements the reference count for an object interface. When the last reference is removed from an object, the object will be destroyed.

Syntax

```
ULONG        Release();
```

Return Values

Value	Description
Count	New reference count – for debug purposes only.

Section 2 - DeckLink API

2.1 Using the DeckLink API in a project

The supplied sample applications provide examples of how to include the DeckLink API in a project on each supported platform.

To use the DeckLink API in your project, one or more files need to be included:

Windows	DeckLink X.Y\Win\Include\DeckLinkAPI.idl
macOS	DeckLink X.Y/Mac/Include/DeckLinkAPI.h
	DeckLink X.Y/Mac/Include/DeckLinkAPIDispatch.cpp
Linux	DeckLink X.Y/Linux/Include/DeckLinkAPI.h
	DeckLink X.Y/Linux/Include/DeckLinkAPIDispatch.cpp

You can also include the optional header file “DeckLinkAPIVersion.h”. It defines two macros containing the SDK version numbers which can be used at runtime by your application to compare the version of the DeckLink API it is linked to with the version of the SDK used at compile time.

2.2 Sandboxing support on macOS

The DeckLink API can be accessed from a sandboxed applications if the following requirements are met:

- Application is built against macOS 10.7 or later
- Ensure “Enable App sandboxing” is ticked in your application’s Xcode project,
- Ensure you have selected a valid code signing identity,
- Insert the following property into your application’s entitlements file:

Refer to the Sandboxed Signal Generator target in the SignalGenerator sample application in the SDK.

Key	Type	Value
com.apple.security.temporary-exception.mach-lookup.global-name	String	com.blackmagic-design.desktopvideo.**DeckLinkHardwareXPCService**

Further information can be found in the App Sandbox Design Guide available on Apple’s Mac Developer Library website.

2.3 Accessing DeckLink devices

Most DeckLink API object interfaces are accessed via the **IDeckLinkIterator** object. How a reference to an **IDeckLinkIterator** is obtained varies between platforms depending on their level of support for COM.

2.3.1 Windows

The main entry point to the DeckLink API is the **IDeckLinkIterator** interface.
This interface should be obtained from COM using CoCreateInstance:

```
IDeckLinkIterator *deckLinkIterator = NULL;

CoCreateInstance(CLSID_CDeckLinkIterator, NULL, CLSCTX_ALL,
IID_IDeckLinkIterator, (void**)&deckLinkIterator);
```

On success, **CoCreateInstance** returns an HRESULT of S_OK and deckLinkIterator points to a new **IDeckLinkIterator** object interface.

2.3.2 macOS and Linux

On platforms without native COM support, a C entry point is provided to access an **IDeckLinkIterator** object:

```
IDeckLinkIterator *deckLinkIterator = CreateDeckLinkIteratorInstance();
```

On success, deckLinkIterator will point to a new **IDeckLinkIterator** object interface otherwise it will be set to NULL.

2.4 High level interface

The DeckLink API provides a framework for video & audio streaming which greatly simplifies the task of capturing or playing out video and audio streams. This section provides an overview of how to use these interfaces.

2.4.1 Capture

An application performing a standard streaming capture operation should perform the following steps:

- If desired, enumerate the supported capture video modes by calling **IDeckLinkInput::GetDisplayModeIterator**. For each reported capture mode, call **IDeckLinkInput::DoesSupportVideoMode** to check if the combination of the video mode and pixel format is supported.
- **IDeckLinkInput::EnableVideoInput**
- **IDeckLinkInput::EnableAudioInput**
- **IDeckLinkInput::SetCallback**
- **IDeckLinkInput::StartStreams**
- While streams are running:
  - receive calls to **IDeckLinkInputCallback::VideoInputFrameArrived** with video frame and corresponding audio packet

IDeckLinkInput::StopStreams

Audio may be “pulled” from a separate thread if desired.

If audio is not required, the call to **IDeckLinkInput::EnableAudioInput** may be omitted and the **IDeckLinkInputCallback::VideoInputFrameArrived** callback will receive NULL audio packets.

2.4.2 Playback

An application performing a standard streaming playback operation should perform the following steps:

- **IDeckLinkOutput::DoesSupportVideoMode** to check if the combination of the video mode and pixel format is supported.
- **IDeckLinkOutput::EnableVideoOutput**
- **IDeckLinkOutput::EnableAudioOutput**
- **IDeckLinkOutput::SetScheduledFrameCompletionCallback**
- **IDeckLinkOutput::SetAudioCallback**
- **IDeckLinkOutput::BeginAudioPreroll**
- While more frames or audio need to be pre-rolled:
  - **IDeckLinkOutput::ScheduleVideoFrame**
  - Return audio data from **IDeckLinkAudioOutputCallback::RenderAudioSamples**
  - When audio preroll is complete, call **IDeckLinkOutput::EndAudioPreroll**
- **IDeckLinkOutput::StartScheduledPlayback**
- While playback is running:
  - Schedule more video frames from **IDeckLinkVideoOutputCallback::ScheduledFrameCompleted**
  - Schedule more audio from **IDeckLinkAudioOutputCallback::RenderAudioSamples**

If audio is not required, the call to **IDeckLinkOutput::EnableAudioOutput, IDeckLinkOutput::SetAudioCallback** and **IDeckLinkOutput::BeginAudioPreroll** may be omitted.

If pre-roll is not required initial **IDeckLinkOutput::ScheduleVideoFrame** calls and the call to **IDeckLinkOutput::BeginAudioPreroll** and **IDeckLinkOutput::EndAudioPreroll** may be omitted.

2.4.3 3D Functionality

3D (dual-stream) capture and playback is supported by certain DeckLink devices such as the DeckLink 4K Extreme. The 3D functionality is only available over HDMI or SDI, where Channel A and Channel B represent the left and right eyes. The 3D packing must be manually set when connecting to pre-HDMI 1.4 devices. When capturing from an HDMI 1.4 compliant source, the 3D packing format will automatically detected, and cannot be overridden. When outputting to an HDMI 1.4 compliant device / monitor, the packing format will be adjusted according to the device / monitor's capabilities, but can be manually changed. Refer to the **IDeckLinkConfiguration** Interface and **BMDVideo3DPackingFormat** sections for more information on getting and setting the packing format.

Note: Automatic mode detection is not available for UHD and DCI 4K 3D dual-link SDI modes.

2.4.3.1 3D Capture

An application performing a streaming 3D capture operation should perform the following steps:

- If desired, enumerate the supported capture video modes by calling **IDeckLinkInput::GetDisplayModeIterator**. For each reported capture mode, check for the presence of the **bmdDisplayModeSupports3D** flag in the return value of **IDeckLinkDisplayMode::GetFlag** indicating that this mode is supported for 3D capture. Call **IDeckLinkInput::DoesSupportVideoMode** with the **bmdVideoInputDualStream3D** flag to check if the combination of the video mode and pixel format is supported.
- Call **IDeckLinkInput::EnableVideoInput** with the **bmdVideoInputDualStream3D** flag.
- **IDeckLinkInput::EnableAudioInput**
- **IDeckLinkInput::SetCallback**
- **IDeckLinkInput::StartStreams**
- While streams are running:
  - Receive calls to IDeckLinkInputCallback::VideoInputFrameArrived with left eye video frame and corresponding audio packet.

  Inside the callback:

  - Call **IDeckLinkVideoInputFrame::QueryInterface** with
  - **IID_IDeckLinkVideoFrame3DExtensions**

  - **IDeckLinkVideoFrame3DExtensions::GetFrameForRightEye**
    The returned frame object must be released by the caller when no longer required.
- **IDeckLinkInput::StopStreams**

2.4.3.2 3D Playback

To support 3D playback, your application must provide the API with a video frame object which implements the **IDeckLinkVideoFrame** interface and returns a valid object implementing the **IDeckLinkVideoFrame3DExtensions** interface when its **QueryInterface** method is called with **IID_IDecklinkVideoFrame3DExtensions**. This can be achieved by providing your own class which:

- subclasses both **IDeckLinkVideoFrame** and **IDeckLinkVideoFrame3DExtensions** interfaces
- returns a pointer to itself (cast to **IDeckLinkVideoFrame3DExtensions**) when its **QueryInterface** method is called with **IID_IDeckLinkVideoFrame3DExtensions**.
- implements all the methods in the **IDeckLinkVideoFrame** and **IDeckLinkVideoFrame3DExtensions** classes.

An application performing a streaming 3D playback operation should perform the following steps:

- Check if 3D is supported for the desired video mode with **IDeckLinkOutput::DoesSupportVideoMode** called with **bmdVideoOutputDualStream3D**.
- Call **IDeckLinkOutput::EnableVideoOutput** with the **bmdVideoOutputDualStream3D** flag set.
- **IDeckLinkOutput::EnableAudioOutput**
- **IDeckLinkOutput::SetScheduledFrameCompletionCallback**
- **IDeckLinkOutput::SetAudioCallback**
- **IDeckLinkOutput::BeginAudioPreroll**
- While more frames or audio need to be pre-rolled:
  - Create a video frame object that subclasses **IDeckLinkVideoFrame** and **IDeckLinkVideoFrame3DExtensions** as explained above.
  - **IDeckLinkOutput::ScheduleVideoFrame**
  - Return audio data from **IDeckLinkAudioOutputCallback::RenderAudioSamples** When audio preroll is complete, call **IDeckLinkOutput::EndAudioPreroll**
- **IDeckLinkOutput::StartScheduledPlayback**
- While playback is running:
  - Schedule more video frames from **IDeckLinkVideoOutputCallback::ScheduledFrameCompleted**
  - Schedule more audio from **IDeckLinkAudioOutputCallback::RenderAudioSamples**

If audio is not required, the call to **IDeckLinkOutput::EnableAudioOutput**, **IDeckLinkOutput::SetAudioCallback** and **IDeckLinkOutput::BeginAudioPreroll** may be omitted.

If pre-roll is not required initial **IDeckLinkOutput::ScheduleVideoFrame** calls and the call to **IDeckLinkOutput::BeginAudioPreroll** and **IDeckLinkOutput::EndAudioPreroll** may be omitted.

2.4.4 DeckLink Device Notification

A callback notification can be provided to an application when a Thunderbolt or USB 3.0 based DeckLink device is connected or disconnected.

An application that supports connection notification should perform the following steps:

- Create a callback class that subclasses **IDeckLinkDeviceNotificationCallback** and implements all of its methods. The callback class will be called asynchronously from an API private thread. Create an instance of the callback class.
- Call **IDeckLinkDiscovery::InstallDeviceNotifications** and provide the **IDeckLinkDeviceNotificationCallback** object.
- **IDeckLinkDeviceNotificationCallback::DeckLinkDeviceArrived** is called for all currently-connected devices.
- When a DeckLink device is connected after the initial reporting of devices then **IDeckLinkDeviceNotificationCallback::DeckLinkDeviceArrived** will be called.
- When a DeckLink device is removed, **IDeckLinkDeviceNotificationCallback::DeckLinkDeviceRemoved** is called on an API-private thread.
- Before the application exits, call **IDeckLinkDiscovery::UninstallDeviceNotifications**.

2.4.5 Streaming Encoder

Streaming encoder functionality is supported by certain DeckLink devices such as the H.264 Pro Recorder. Uncompressed video and audio streams may be encoded into a compressed bitstream and made available to suitable applications involving compressed video and audio.

2.4.5.1 Streaming Encoder Capture

An application performing a typical streaming encoder capture operation should perform the following steps:

- Enumerate the preset video encoding modes by calling **IBMDStreamingDeviceInput::GetVideoEncodingModePresetIterator**.
  For each reported video encoding mode call **IBMDStreamingDeviceInput::GetCurrentDetectedVideoInputMode** and **IBMDStreamingDeviceInput::DoesSupportVideoEncodingMode** to check if the current video input mode and video encoding mode are supported.
- If desired, call **IBMDStreamingVideoEncodingMode::CreateMutableVideoEncodingMode** to change the encoder bitrate or other encoder settings.
- **IBMDStreamingDeviceInput::SetVideoEncodingMode**
- **IBMDStreamingDeviceInput::SetCallback**
- **IBMDStreamingDeviceInput::StartCapture**
- While capture is running:
  - receive calls to **IBMDStreamingH264InputCallback::MPEG2TSPacketArrived** with MPEG transport stream data to process both compressed video and audio
  - alternatively, receive calls to **IBMDStreamingH264InputCallback::H264NALPacketArrived** and **IBMDStreamingH264InputCallback::H264AudioPacketArrived** to process compressed video and audio data separately
- **IBMDStreamingDeviceInput::StopCapture**

2.4.6 Automatic Mode Detection

The automatic mode detection feature will notify an application when a property of the video input signal changes. This feature is supported on certain DeckLink devices. For an example of using automatic mode detection, please refer the AutomaticModeDetection sample in the DeckLink SDK.

To use this feature please refer to the following steps:

- Call **IDeckLinkAttributes::GetFlag** with the **BMDDeckLinkSupportsInputFormatDetection** flag to check that the DeckLink hardware supports the automatic format detection feature.
- Create a callback class that subclasses from **IDeckLinkInputCallback** and implements all of its methods.
  The **IDeckLinkInputCallback::VideoInputFormatChanged** method will be called when a change in the property of the video signal has been detected.
- Install a callback by calling **IDeckLinkInput::SetCallback** and referencing an instance of your callback class.
- Call **IDeckLinkInput::EnableVideoInput** with an initial video mode and pixel format and set the **bmdVideoInputEnableFormatDetection** flag.
- Call **IDeckLinkInput::EnableAudioInput**.
- Call **IDeckLinkInput::StartStreams** to begin capture.
- While the input streams are running:
  - If a change in a property of the input video signal is detected then **IDeckLinkInputCallback::VideoInputFormatChanged** will be called in your callback object with the new video properties provided in the parameters.
  - If the video mode or pixel format has changed, then the following sequence could be used to restart capture with the new settings:

    IDeckLinkInput::PauseStreams

    Call **IDeckLinkInput::EnableVideoInput** with the detected video mode and pixel format.

    IDeckLinkInput::FlushStreams

    IDeckLinkInput::StartStreams
- Call **IDeckLinkInput::StopStreams** to stop capture.
- Call **IDeckLinkInput::DisableVideoInput**
- Call **IDeckLinkInput::DisableAudioInput**

2.4.7 Ancillary Data functionality

The capture or output of vertical ancillary data (VANC) is supported by certain DeckLink device models. The lines of VANC that are accessible are dependent upon the model of the DeckLink device. Currently horizontal ancillary data (HANC) access is not supported.

2.4.7.1 VANC Capture

An application performing VANC data capture should perform the following steps:

- **IDeckLinkInput::EnableVideoInput**
  The pixel format that is specified will apply to both active picture and ancillary data with non-4K DeckLink devices.
  When capturing ancillary data with a 4K DeckLink device, the ancillary data will always be in the 10-bit YUV pixel format.
- **IDeckLinkInput::EnableAudioInput**
- **IDeckLinkInput::SetCallback**
- **IDeckLinkInput::StartStreams**
- While streams are running:

  - Receive calls to **IDeckLinkInputCallback::VideoInputFrameArrived**

  Inside the callback:

  - Call **IDeckLinkVideoFrame::QueryInterface** with **IID_IDeckLinkVideoFrameAncillaryPackets.**

    - As the **IDeckLinkVideoFrameAncillaryPackets** object has a reference to the **IDeckLinkVideoFrame** input frame, ensure that it is released in a timely manner, otherwise the capture will run out of available frames.

  - If the DID/SDID for the ancillary packet is known, then call **IDeckLinkVideoFrameAncillaryPackets::GetFirstPacketByID.**

    - Check that S_OK is returned to confirm an ancillary packet with matching DID/SDID is found.

  - Otherwise, enumerate the ancillary packets in the video frame by calling **IDeckLinkVideoFrameAncillaryPackets::GetPacketIterator.**

  - **IDeckLinkAncillaryPacket::GetBytes**

  - The output packet payload will be converted to the requested **BMDAncillaryPacketFormat**

2.4.7.2 VANC Output

An application performing VANC output should perform the following steps.

For an example of performing VANC output, please refer the VancOutput sample in the DeckLink SDK.

- Call **IDeckLinkOutput::EnableVideoOutput** with the **bmdVideoOutputVANC** flag set.
- Create a ancillary packet object that subclasses **IDeckLinkAncillaryPacket**, implementing all methods of the **IDeckLinkAncillaryPacket** class.
- **IDeckLinkAncillaryPacket::GetBytes**

  - Implement to provide pointer to packet data in playback operation.

  - The packet payload data shall be implemented with at least one **BMDAncillaryPacketFormat.** The driver will automatically convert to the correct format on output.

- **IDeckLinkOutput::CreateVideoFrame**
- Call **IDeckLinkVideoFrame::QueryInterface** with **IID_IDeckLinkVideoFrameAncillaryPackets**.
  - As the **IDeckLinkVideoFrameAncillaryPackets** object has a reference to the **IDeckLinkVideoFrame** input frame, ensure that it is released in a timely manner, otherwise the playback will run out of available frames.
- **IDeckLinkVideoFrameAncillaryPackets::AttachPacket**
  - Attach ancillary packet to video frame for playback.
- **IDeckLinkOutput::ScheduleVideoFrame**
- **IDeckLinkOutput::StartScheduledPlayback**

2.4.8 Keying

Alpha keying allows an application to either superimpose a key frame over an incoming video feed (internal keying) or to send fill and key to an external keyer (external keying). The alpha keying functionality is supported on certain DeckLink models.

For an example of using the keying functionality please refer to GdiKeyer sample application in the DeckLink SDK.

An application performing keying should use the following steps:

- Detect the keying capabilities of the hardware
  - Call **IDeckLinkAttributes::GetFlag** using **BMDDeckLinkSupportsInternalKeying** or **BMDDeckLinkSupportsExternalKeying** to determine internal / external keying support
  - Call **IDeckLinkAttributes::GetFlag** using **BMDDeckLinkSupportsHDKeying** to determine if keying is supported with high definition video modes.
- Create video frames with pixel formats that have alpha channels (such as **bmdFormat8BitARGB** or **bmdFormat8BitBGRA**).
- **IDeckLinkOutput::EnableVideoOutput**
- Call **IDeckLinkKeyer::Enable** with FALSE for internal keying or TRUE for external keying
- Set a fixed level of blending using **IDeckLinkKeyer::SetLevel**
  Alternatively set ramp up or down blending using **IDeckLinkKeyer::RampUp** or **IDeckLinkKeyer::RampDown**
  The level of blending of each pixel will depend on the value in the alpha channel and the keying level setting.
- **IDeckLinkOutput::SetScheduledFrameCompletionCallback**
- Pre-roll video frames using **IDeckLinkOutput::ScheduleVideoFrame**
- **IDeckLinkOutput::StartScheduledPlayback**
- While playback is running schedule video frames from **IDeckLinkVideoOutputCallback::ScheduledFrameCompleted**
- When playback has finished:
  - **IDeckLinkKeyer::Disable**
  - **IDeckLinkOutput::DisableVideoOutput**

2.4.9 Timecode/Timecode user bits

The capture and output of VITC and RP188 timecodes are supported on certain DeckLink models. VITC timecodes are only supported with SD video modes. On non-4K DeckLink devices, RP188 timecodes are only supported with HD video modes.

To use this feature please refer to the following points:

2.4.9.1 Timecode Capture

An application performing timecode capture should perform the following steps. For an example of timecode capture please refer to the CapturePreview sample application in the DeckLink SDK.

- **IDeckLinkInput::EnableVideoInput**
- **IDeckLinkInput::EnableAudioInput**
- **IDeckLinkInput::SetCallback**
- **IDeckLinkInput::StartStreams**
- While streams are running:
  - Receive calls to **IDeckLinkInputCallback::VideoInputFrameArrived** with video frame and corresponding audio packet
  - Call **IDeckLinkVideoInputFrame::GetTimecode**
  - **IDeckLinkTimecode::GetFlags**
  - **IDeckLinkTimecode::GetTimecodeUserBits**
- **IDeckLinkInput::StopStreams**
- **IDeckLinkInput::DisableVideoInput**

2.4.9.2 Timecode Output

An application performing timecode output should perform the following steps. For an example of timecode output please refer to the Linux SignalGenerator sample application in the DeckLink SDK.

- Call **IDeckLinkOutput::EnableVideoOutput** with either **bmdVideoOutputVITC** or **bmdVideoOutputRP188**
- **IDeckLinkOutput::EnableAudioOutput**
- **IDeckLinkOutput::SetScheduledFrameCompletionCallback**
- **IDeckLinkOutput::SetAudioCallback**
- **IDeckLinkOutput::BeginAudioPreroll**
- While more frames or audio need to be pre-rolled:
  - Create video frames with **IDeckLinkOutput::CreateVideoFrame**
  - Set the timecode into the frame with **IDeckLinkMutableVideoFrame::SetTimecode** or **IDeckLinkMutableVideoFrame::SetTimecodeFromComponents**
  - **IDeckLinkOutput::ScheduleVideoFrame**
  - Return audio data from **IDeckLinkAudioOutputCallback::RenderAudioSamples**
  - When audio preroll is complete, call **IDeckLinkOutput::EndAudioPreroll**
- **IDeckLinkOutput::StartScheduledPlayback**
- While playback is running:
  - Create video frames and set the timecode.
  - Schedule more video frames from **IDeckLinkVideoOutputCallback::ScheduledFrameCompleted**
  - Schedule more audio from **IDeckLinkAudioOutputCallback::RenderAudioSamples**
- **IDeckLinkOutput::StopScheduledPlayback**
- **IDeckLinkOutput::DisableVideoOutput**

2.4.10 H.265 Capture

Certain DeckLink devices support encoded (e.g. H.265) capture in addition to regular uncompressed capture.

Note that the Encoded Capture interface is distinct from the H.264 only 'Streaming Encoder' interface.

2.4.10.1 Encoded Capture

An application performing an encoded capture operation should perform the following steps:

- Obtain a reference to the **IDeckLinkEncoderInput** interface from **IDeckLinkInput** via **QueryInterface**
- If desired, enumerate the supported encoded capture video modes by calling **IDeckLinkEncoderInput::GetDisplayModeIterator.**
  For each reported capture mode, call **IDeckLinkEncoderInput::DoesSupportVideoMode** to check if the combination of the video mode and pixel format is supported.
- **IDeckLinkEncoderInput::EnableVideoInput**
- **IDeckLinkEncoderInput::EnableAudioInput**
- **IDeckLinkEncoderInput::SetCallback**
- **IDeckLinkEncoderInput::StartStreams**
- While streams are running:
  - receive calls to **IDeckLinkEncoderInputCallback::VideoPacketArrived** with encoded video packets
  - receive calls to **IDeckLinkEncoderInputCallback::AudioPacketArrived** with audio packets
- **IDeckLinkInput::StopStreams**

If audio is not required, the call to **IDeckLinkEncoderInput::EnableAudioInput** may be omitted and the **IDeckLinkEncoderInputCallback::AudioPacketArrived** callback will not be called.

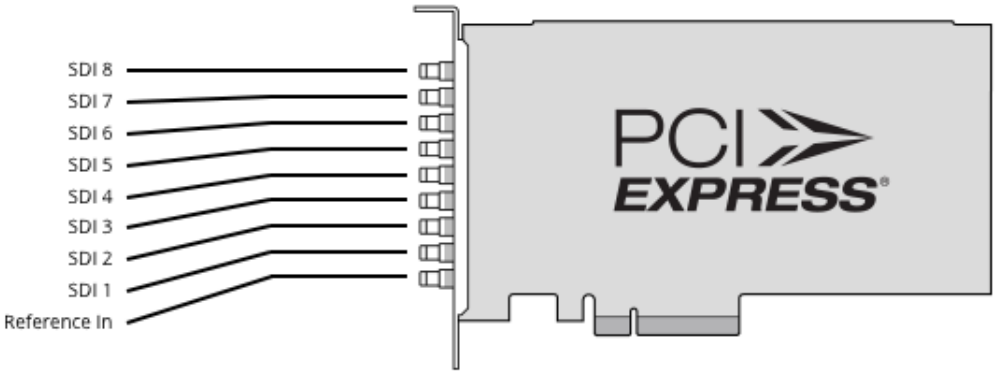
2.4.11 Configurable duplex mode

Certain DeckLink devices such as the DeckLink Quad 2 and the DeckLink Duo 2 support configuration of the duplex mode of individual sub-devices.

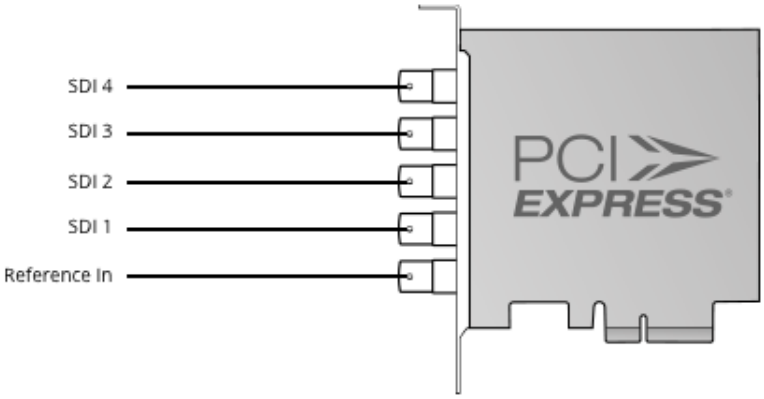
For these DeckLink devices, sub-devices are grouped together in pairs.

- A sub-device configured as full-duplex will use two connectors, which allows simultaneous capture and playback, internal keying, and fill & key scenarios.
- A half-duplex sub-device will use a single connector as an individual capture or playback channel.
- A sub-device pair shares two connectors, therefore a sub-device configured as full-duplex will cause the paired sub-device to become inactive.

The table and illustration below demonstrate the pairing between sub-devices, and how the relationship to physical connectors varies in some common duplex mode configurations.



Sub-device index	Connector mapping in Half-Duplex	Connector mapping in Full-Duplex	Configurable Duplex Mode	Paired sub-device index
0	SDI 1	SDI 1 (in/key) & SDI 2 (out/fill)	Yes	4
1	SDI 3	SDI 3 (in/key) & SDI 4 (out/fill)	Yes	5
2	SDI 5	SDI 5 (in/key) & SDI 6 (out/fill)	Yes	6
3	SDI 7	SDI 7 (in/key) & SDI 8 (out/fill)	Yes	7
4	SDI 2	-		0
5	SDI 4	-		1
6	SDI 6	-		2
7	SDI 8	-		3



Sub-device index	Connector mapping in Half-Duplex	Connector mapping in Full-Duplex	Configurable Duplex Mode	Paired sub-device index
0	SDI 1	SDI 1 (in/key) & SDI 2 (out/fill)	Yes	2
1	SDI 3	SDI 3 (in/key) & SDI 4 (out/fill)	Yes	3
2	SDI 2	-		0
3	SDI 4	-		1

Sub-device and Paired sub-device index reference the value of the BMDDeckLinkSubDeviceIndex attribute. The Paired sub-device is identified by the value of the BMDDeckLinkPairedDevicePersistentID attribute (see IDeckLinkAttributes interface for details).

For example, as can be seen in the tables above:

- DeckLink Quad 2 sub-device 0 is paired with sub-device 4. DeckLink Duo 2 sub-device 0 is pared with sub-device 2.
- Duplex mode is only configurable on the first of each pair, sub-device 0.
- Duplex mode configured on sub-device 0 also affects the paired sub-device.
- Connector SDI 2 is shared between sub-device 0 and its paired sub-device.
- Configuring sub-device 0 as half-duplex causes sub-device 0 to use connector SDI 1 and the paired sub-device to use connector SDI 2.
- Alternatively, configuring sub-device 0 as full-duplex causes sub-device 0 to use connectors SDI 1 and SDI 2, and the paired sub-device to use none.

2.4.11.1 Configuring duplex mode

An application may perform the following steps to configure the Duplex mode:

Duplex mode can only be configured on the first sub-device in a pair.

- Call **IDeckLinkAttributes::GetFlag** with the **BMDDeckLinkSupportsDuplexModeConfiguration** flag to check that the DeckLink sub-device supports duplex mode configuration.
  - If desired, call **IDeckLinkAttributes::GetInt** with the **BMDDeckLinkPairedDevicePersistentID** attribute to identify the paired sub-device which will be affected by the duplex mode configuration change, or which sub-device can be used to change the duplex mode for the current device.
- Call **IDeckLinkConfiguration::SetInt** with the **bmdDeckLinkConfigDuplexMode** configuration ID, along with the desired **BMDDuplexMode** setting to change the Duplex Mode.

2.4.12 HDR Metadata

HDR Metadata capture and playback is supported by certain DeckLink devices such as the DeckLink 4K Extreme 12G. An application performing capture or playback with HDR Metadata should first verify support of this feature by calling **IDeckLinkAttribute::GetFlag** with attribute **BMDDeckLinkSupportsHDRMetadata**. The **IDeckLinkVideoFrameMetadataExtensions** object interface provides methods to query metadata associated with a video frame.

2.4.12.1 HDR Metadata Capture

An application performing capture of video frames with HDR Metadata should perform the following steps:

- **IDeckLinkInput::EnableVideoInput**
- **IDeckLinkInput::SetCallback**
- **IDeckLinkInput::StartStreams**
- While streams are running:
  - Receive calls to IDeckLinkInputCallback::VideoInputFrameArrived

  Inside the callback:
  - Check that video frame has HDR Metadata by ensuring **IDeckLinkVideoFrame::GetFlags** has **bmdFrameContainsHDRMetadata** flag.
  - Call **IDeckLinkVideoInputFrame::QueryInterface** with **IID_IDeckLinkVideoFrameMetadataExtensions**.
  - **IDeckLinkVideoFrameMetadataExtensions::Get*** methods can be called to access HDR Metadata items. See **BMDDeckLinkFrameMetadataID** enumerator for a full list of supported HDR Metadata items.
  - The **IDeckLinkVideoFrameMetadataExtensions** object must be released by the caller when no longer required.

2.4.12.2 HDR Metadata Playback

In order to output HDR metadata, your application must provide the API with a custom video frame object which implements the **IDeckLinkVideoFrame** interface and returns a valid object implementing the **IDeckLinkVideoFrameMetadataExtensions** interface when its **QueryInterface** method is called with **IID_IDeckLinkVideoFrameMetadataExtensions**. This can be achieved by providing your own class which:

- subclasses both **IDeckLinkVideoFrame** and **IDeckLinkVideoFrameMetadataExtensions** interfaces.
- returns a pointer to itself (cast to **IDeckLinkVideoFrameMetadataExtensions**) when its **QueryInterface** method is called with **IID_IDeckLinkVideoFrameMetadataExtensions**.
- implements all the methods in the **IDeckLinkVideoFrame** class.
- specify the HDR metadata items to be queried by implementing methods in the **IDeckLinkVideoFrameMetadataExtensions** class. See **BMDDeckLinkFrameMetadataID** enumerator for a full list of supported HDR Metadata items.
- reveal the presence of HDR Metadata in custom frame by returning flag **bmdFrameContainsHDRMetadata** when video frame flags are queried with **IDeckLinkVideoFrame::GetFlags**

An application performing output with HDR Metadata should perform the following steps:

- **IDeckLinkOutput::EnableVideoOutput**
- **IDeckLinkOutput::SetScheduledFrameCompletionCallback**
- While more frames or audio need to be pre-rolled:
  - Create a custom video frame object that subclasses **IDeckLinkVideoFrame** and **IDeckLinkVideoFrameMetadataExtensions** as explained above.
  - **IDeckLinkOutput::ScheduleVideoFrame**
- **IDeckLinkOutput::StartScheduledPlayback**
- While playback is running:
  - Schedule more custom video frames from **IDeckLinkVideoOutputCallback::ScheduledFrameCompleted**

2.5 Interface Reference

2.5.1 IDeckLinkIterator Interface

The **IDeckLinkIterator** interface is used to enumerate the available DeckLink devices.

A reference to an **IDeckLinkIterator** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateDeckLinkIteratorInstance** on other platforms.

The **IDeckLink** interface(s) returned may be used to access the related interfaces which provide access to the core API functionality.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	**IDeckLinkIterator::Next** returns **IDeckLink** interfaces representing each attached DeckLink device.

Public Member Functions	
Method	**Description**
Next	Returns an **IDeckLink** object interface corresponding to an individual DeckLink device.

2.5.1.1 IDeckLinkIterator::Next method

The **Next** method creates an object representing a physical DeckLink device and assigns the address of the IDeckLink interface of the newly created object to the decklinkInstance parameter.

Syntax

```
HRESULT      Next (IDeckLink **decklinkInstance);
```

Parameters

Name	Direction	Description
`decklinkInstance`	out	Next IDeckLink object interface

Return Values

Value	Description
S_FALSE	No (more) devices found
E_FAIL	Failure
S_OK	Success

2.5.2 IDeckLink Interface

The **IDeckLink** object interface represents a physical DeckLink device attached to the host computer.

IDeckLink object interfaces are obtained from **IDeckLinkIterator**. **IDeckLink** may be queried to obtain the related **IDeckLinkOutput**, **IDeckLinkInput** and **IDeckLinkConfiguration** interfaces.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkIterator`	`IID_IDeckLinkIterator`	**IDeckLinkIterator::Next** returns **IDeckLink** interfaces representing each attached DeckLink device.
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkOutput** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkInput`	`IID_IDeckLinkInput`	An **IDeckLinkInput** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkConfiguration`	`IID_IDeckLinkConfiguration`	An **IDeckLinkConfiguration** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkAttributes`	`IID_IDeckLinkAttributes`	An **IDeckLinkAttributes** object interface may be obtained from **IDeckLink** using **QueryInterface**.
`IDeckLinkKeyer`	`IID_IDeckLinkKeyer`	An **IDeckLinkKeyer** object interface may be obtained from **IDeckLink** using **QueryInterface**.
`IDeckLinkDeck Control`	`IID_IDeckLinkDeckControl`	An **IDeckLinkDeckControl** object may be obtained from **IDeckLink** using **QueryInterface**

Public Member Functions	
Method	**Description**
GetModelName	Method to get DeckLink device model name.
GetDisplayName	Method to get a device name suitable for user interfaces

2.5.2.1 IDeckLink::GetModelName method

The **GetModelName** method can be used to get DeckLink device model name.

Syntax

```
HRESULT      GetModelName (string *modelName);
```

Parameters

Name	Direction	Description
`modelName`	out	Hardware model name. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.2.2 IDeckLink::GetDisplayName method

The **GetDisplayName** method returns a string suitable for display in a user interface. If the device has a custom label specified (see **bmdDeckLinkConfigDeviceInformationLabel**), the label will be used as the display name for the device.

Otherwise, the string is made of the model name (as returned by **GetModelName**) followed by an increasing number (starting from 1) if more than one instance of a device is present in the system. If not, the returned string is simply the model name.

Syntax

```
HRESULT      GetDisplayName (string *displayName);
```

Parameters

Name	Direction	Description
`displayName`	out	The device's display name. This allocated string must be freed by caller when no longer required

Return Values

Value	Description
E_FAIL	Failed to allocate the string
S_OK	Success

2.5.3 IDeckLinkOutput interface

The **IDeckLinkOutput** object interface allows an application to output a video and audio stream from a DeckLink device.

An **IDeckLinkOutput** interface can be obtained from an **IDeckLink** object interface using QueryInterface. If QueryInterface for an output interface is called on an input only device, then QueryInterface will fail and return E_NOINTERFACE.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkOutput** object interface may be obtained from **IDeckLink** using QueryInterface
`IDeckLinkDisplayModeIterator`	`IID_IDeckLinkDisplayModeIterator`	**IDeckLinkOutput::GetDisplayModeIterator** returns an **IDeckLinkDisplayModeIterator** object interface
`IDeckLinkVideoFrame`	`IID_DeckLinkVideoFrame`	**IDeckLinkOutput::CreateVideoFrame** may be used to create a new **IDeckLinkVideoFrame** object interface
`IDeckLinkVideoOutputCallback`	`IID_DeckLinkVideoOutputCallback`	An **IDeckLinkVideoOutputCallback** object interface may be registered with **IDeckLinkOutput::SetScheduledFrameCompletionCallback**
`IDeckLinkAudioOutputCallback`	`IID_DeckLinkAudioOutputCallback`	An **IDeckLinkAudioOutputCallback** object interface may be registered with **IDeckLinkOutput::SetAudioCallback**

Public Member Functions	
Method	**Description**
DoesSupportVideoMode	Check whether a given video mode is supported for output
GetDisplayModeIterator	Get an iterator to enumerate the available output display modes
SetScreenPreviewCallback	Register screen preview callback
EnableVideoOutput	Enable video output
DisableVideoOutput	Disable video output
SetVideoOutputFrameMemoryAllocator	Register custom memory allocator
CreateVideoFrame	Create a video frame
CreateAncillaryData	Create ancillary buffer
DisplayVideoFrameSync	Display a video frame synchronously
ScheduleVideoFrame	Schedule a video frame for display
SetScheduledFrameCompletionCallback	Register completed frame callback
GetBufferedVideoFrameCount	Gets number of frames queued.
EnableAudioOutput	Enable audio output
DisableAudioOutput	Disable audio output
WriteAudioSamplesSync	Play audio synchronously
BeginAudioPreroll	Start pre-rolling audio
EndAudioPreroll	Stop pre-rolling audio
ScheduleAudioSamples	Schedule audio samples for play-back
GetBufferedAudioSampleFrameCount	Returns the number of audio sample frames currently buffered for output
FlushBufferedAudioSamples	Flush buffered audio
SetAudioCallback	Register audio output callback
StartScheduledPlayback	Start scheduled playback
StopScheduledPlayback	Stop scheduled playback
GetScheduledStreamTime	Returns the elapsed time since scheduled playback began.
IsScheduledPlaybackRunning	Determine if the video output scheduler is running
GetHardwareReferenceClock	Get scheduling time
GetReferenceStatus	Provides reference genlock status

2.5.3.1 IDeckLinkOutput::DoesSupportVideoMode method

The **DoesSupportVideoMode** method indicates whether a given display mode is supported on output. Modes may be supported, unsupported or supported with conversion. If the requested video mode cannot be output then **bmdDisplayModeSupportedWithConversion** will be returned and the video will be converted into a supported video mode indicated by resultDisplayMode.

Note: If a pixel format is not natively supported in the card's hardware it will be converted by software.

Syntax

```
HRESULT     DoesSupportVideoMode
            (BMDDisplayMode displayMode, BMDPixelFormat pixelFormat,
            BMDVideoOutputFlags flags, BMDDisplayModeSupport *support,
            IDeckLinkDisplayMode **resultDisplayMode);
```

Parameters

Name	Direction	Description
`displayMode`	in	Display mode to check
`pixelFormat`	in	Pixel format to check (0 for any)
`flags`	in	Video output flags (see **BMDVideoOutputFlags** for details).
`support`	out	Video output mode supported result.
`resultDisplayMode`	out	If this parameter is not NULL, an **IDeckLinkDisplayMode** object representing the given displayMode is returned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.2 IDeckLinkOutput::IsScheduledPlaybackRunning method

The **IsScheduledPlaybackRunning** method is called to determine if the driver's video output scheduler is currently active.

Syntax

```
HRESULT      IsScheduledPlaybackRunning (boolean *active)
```

Parameters

Name	Direction	Description
`active`	out	Active status of driver video output scheduler

Return Values

Value	Description
E_INVALIDARG	Parameter active status variable is NULL
E_FAIL	Failure
S_OK	Success

2.5.3.3 IDeckLinkOutput::GetDisplayModeIterator method

The **GetDisplayModeIterator** method returns an iterator which enumerates the available display modes.

Syntax

```
HRESULT      GetDisplayModeIterator
             (IDeckLinkDisplayModeIterator **iterator);
```

Parameters

Name	Direction	Description
`iterator`	out	Display mode iterator

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.4 IDeckLinkOutput::SetScreenPreviewCallback method

The **SetScreenPreviewCallback** method is called to register an instance of an **IDeckLinkScreenPreviewCallback** object. The registered object facilitates the updating of an on-screen preview of a video stream being played.

Syntax

```
HRESULT     SetScreenPreviewCallback
            (IDeckLinkScreenPreviewCallback *previewCallback)
```

Parameters

Name	Direction	Description
`previewCallback`	in	The IDeckLinkScreenPreview object to be registered.

Return Values

Value	Description
E_OUTOFMEMORY	Unable to create kernel event (Windows only)
E_FAIL	Failure
S_OK	Success

2.5.3.5 IDeckLinkOutput::EnableVideoOutput method

The **EnableVideoOutput** method enables video output. Once video output is enabled, frames may be displayed immediately with **DisplayVideoFrameSync** or scheduled with **ScheduleVideoFrame**.

Syntax

```
HRESULT     EnableVideoOutput
            (BMDDisplayMode displayMode, BMDVideoOutputFlags flags);
```

Parameters

Name	Direction	Description
`displayMode`	in	Display mode for video output
`flags`	in	Flags to control ancillary data and video output features.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Unable to access the hardware
E_OUTOFMEMORY	Unable to create a new frame

2.5.3.6 IDeckLinkOutput::DisableVideoOutput method

The **DisableVideoOutput** method disables video output.

Syntax

```
HRESULT      DisableVideoOutput ();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.7 IDeckLinkOutput:: SetVideoOutputFrameMemoryAllocator method

The **SetVideoOutputFrameMemoryAllocator** method sets a custom memory allocator for video frame allocations during playback.
The use of a custom memory allocator is optional.

Syntax

```
HRESULT      SetVideoOutputFrameMemoryAllocator
             (IDeckLinkMemoryAllocator *theAllocator);
```

Parameters

Name	Direction	Description
`theAllocator`	in	Allocator object with an **IDeckLinkMemoryAllocator** interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.8 IDeckLinkOutput::CreateVideoFrame method

The **CreateVideoFrame** method creates a video frame for output (see **IDeckLinkMutableVideoFrame** for more information).

Syntax

```
HRESULT     CreateVideoFrame
            (long width, long height, long rowBytes,
            BMDPixelFormat pixelFormat, BMDFrameFlags flags,
            IDeckLinkMutableVideoFrame **outFrame);
```

Parameters

Name	Direction	Description
`width`	in	frame width in pixels
`height`	in	frame height in pixels
`rowBytes`	in	bytes per row
`pixelFormat`	in	pixel format
`flags`	in	frame flags
`outFrame`	out	newly created video frame

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.9 IDeckLinkOutput::CreateAncillaryData method

The **CreateAncillaryData** method creates an ancillary buffer that can be attached to an **IDeckLinkMutableVideoFrame**.

Syntax

```
HRESULT     CreateAncillaryData
            (BMDPixelFormat pixelFormat,
            IDeckLinkVideoFrameAncillary** outBuffer);
```

Parameters

Name	Direction	Description
`pixelFormat`	in	Pixel format for ancillary data
`outBuffer`	out	New video frame ancillary buffer

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Video output is not enabled.

2.5.3.10 IDeckLinkOutput::DisplayVideoFrameSync method

The **DisplayVideoFrameSync** method is used to provide a frame to display as the next frame output. It should not be used during scheduled playback.

Video output must be enabled with **EnableVideoOutput** before frames can be displayed.

Syntax

```
HRESULT     DisplayVideoFrameSync (IDeckLinkVideoFrame *theFrame);
```

Parameters

Name	Direction	Description
`theFrame`	in	frame to display – after call return, the frame may be released

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	The video output is not enabled.
E_INVALIDARG	The frame attributes are invalid.

2.5.3.11 IDeckLinkOutput::ScheduleVideoFrame method

The **ScheduleVideoFrame** method is used to schedule a frame for asynchronous playback at a specified time.

Video output must be enabled with **EnableVideoOutput** before frames can be displayed. Frames may be scheduled before calling **StartScheduledPlayback** to preroll. Once playback is initiated, new frames can be scheduled from **IDeckLinkVideoOutputCallback**.

Syntax

```
HRESULT     ScheduleVideoFrame
            (IDeckLinkVideoFrame *theFrame, BMDTimeValue displayTime,
            BMDTimeValue displayDuration, BMDTimeScale timeScale);
```

Parameters

Name	Direction	Description
`theFrame`	in	frame to display
`displayTime`	in	time at which to display the frame in timeScale units
`displayDuration`	in	duration for which to display the frame in timeScale units
`timeScale`	in	time scale for displayTime and displayDuration

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	The video output is not enabled.
E_INVALIDARG	The frame attributes are invalid.
E_OUTOFMEMORY	Too many frames are already scheduled

2.5.3.12 IDeckLinkOutput:: SetScheduledFrameCompletionCallback method

The **SetScheduledFrameCompletionCallback** method configures a callback which will be called when each scheduled frame is completed.

Syntax

```
HRESULT     SetScheduledFrameCompletionCallback
            (IDeckLinkVideoOutputCallback *theCallback);
```

Parameters

Name	Direction	Description
`theCallBack`	in	Callback object implementing the **IDeckLinkVideoOutputCallback** object interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.13 IDeckLinkOutput::GetBufferedVideoFrameCount method

The **GetBufferedVideoFrameCount** method gets the number of frames queued.

Syntax

```
HRESULT     GetBufferedVideoFrameCount
            (uint32_t *bufferedFrameCount);
```

Parameters

Name	Direction	Description
`bufferedFrameCount`	out	The frame count.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.14 IDeckLinkOutput::EnableAudioOutput method

The **EnableAudioOutput** method puts the hardware into a specified audio output mode. Once audio output is enabled, sample frames may be output immediately using **WriteAudioSamplesSync** or as part of scheduled playback using **ScheduleAudioSamples**.

Syntax

```
HRESULT     EnableAudioOutput
            (BMDAudioSampleRate sampleRate,
            BMDAudioSampleType sampleType,
            uint32_t channelCount,
            BMDAudioOutputStreamType streamType);
```

Parameters

Name	Direction	Description
`sampleRate`	in	Sample rate to output
`sampleType`	in	Sample type to output
`channelCount`	in	Number of audio channels to output – only 2, 8 or 16 channel output is supported.
`streamType`	in	Type of audio output stream.

Return Values

Value	Description
E_FAIL	Failure
E_INVALIDARG	Invalid number of channels requested
S_OK	Success
E_ACCESSDENIED	Unable to access the hardware or audio output not enabled.
E_OUTOFMEMORY	Unable to create internal object

2.5.3.15 IDeckLinkOutput::DisableAudioOutput method

The **DisableAudioOutput** method disables the hardware audio output mode.

Syntax

```
HRESULT     DisableAudioOutput ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.16 IDeckLinkOutput::WriteAudioSamplesSync method

The **WriteAudioSamplesSync** method is used to play audio sample frames immediately. Audio output must be configured with **EnableAudioOutput**. **WriteAudioSamplesSync** should not be called during scheduled playback.

Syntax

```
HRESULT     WriteAudioSamplesSync
            (void *buffer, uint32_t sampleFrameCount,
            uint32_t *sampleFramesWritten);
```

Parameters

Name	Direction	Description
`buffer`	in	Buffer containing audio sample frames. Audio channel samples must be interleaved into a sample frame and sample frames must be contiguous.
`sampleFrameCount`	in	Number of sample frames available
`sampleFramesWritten`	out	Actual number of sample frames queued

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.17 IDeckLinkOutput::BeginAudioPreroll method

The **BeginAudioPreroll** method requests the driver begin polling the registered **IDeckLinkAudioOutputCallback::RenderAudioSamples** object interface for audio-preroll.

Syntax

```
HRESULT     BeginAudioPreroll ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.18 IDeckLinkOutput::EndAudioPreroll method

The **EndAudioPreroll** method requests the driver stop polling the registered **IDeckLinkAudioOutputCallback** object interface for audio-preroll.

Syntax

```
HRESULT      EndAudioPreroll ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.19 IDeckLinkOutput::ScheduleAudioSamples method

The **ScheduleAudioSamples** method is used to provide audio sample frames for scheduled playback. Audio output must be enabled with **EnableAudioOutput** before frames may be scheduled.

Syntax

```
HRESULT      ScheduleAudioSamples
             (void *buffer, uint32_t sampleFrameCount,
             BMDTimeValue streamTime, BMDTimeScale timeScale,
             uint32_t *sampleFramesWritten);
```

Parameters

Name	Direction	Description
`buffer`	in	Buffer containing audio sample frames. Audio channel samples must be interleaved into a sample frame and sample frames must be contiguous.
`sampleFrameCount`	in	Number of sample frames available
`streamTime`	in	Time for audio playback in units of timeScale. To queue samples to play back immediately after currently buffered samples both streamTime and timeScale may be set to zero when using **bmdAudioOutputStreamContinuous**.
`timeScale`	in	Time scale for the audio stream.
`sampleFramesWritten`	out	Actual number of sample frames scheduled

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Either audio output has not been enabled or an audio sample write is in progress.
E_INVALIDARG	No timescale has been provided. A timescale is necessary as the audio packets are time-stamped.

2.5.3.20 IDeckLinkOutput:: GetBufferedAudioSampleFrameCount method

The **GetBufferedAudioSampleFrameCount** method returns the number of audio sample frames currently buffered for output. This method may be used to determine how much audio is currently buffered before scheduling more audio with **ScheduleAudioSamples**.

Syntax

```
HRESULT      GetBufferedAudioSampleFrameCount
             (uint32_t *bufferedSampleFrameCount)
```

Parameters

Name	Direction	Description
`bufferedSampleFrameCount`	out	Number of audio frames currently buffered.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.21 IDeckLinkOutput::FlushBufferedAudioSamples method

The **FlushBufferedAudioSamples** method discards any buffered audio sample frames.

FlushBufferedAudioSamples should be called when changing playback direction. Buffered audio is implicitly flushed when stopping audio playback with **StopScheduledPlayback** or **DisableAudioOutput**.

Syntax

```
HRESULT      FlushBufferedAudioSamples ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.22 IDeckLinkOutput::SetAudioCallback method

The **SetAudioCallback** method configures a callback which will be called regularly to allow the application to queue audio for scheduled playback.

Use of this method is optional – audio may alternately be queued from **IDeckLinkVideoOutputCallback::ScheduledFrameCompleted**.

Syntax

```
HRESULT     SetAudioCallback
            (IDeckLinkAudioOutputCallback *theCallback);
```

Parameters

Name	Direction	Description
`theCallBack`	in	callback object implementing the **IDeckLinkAudioOutputCallback** object interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.23 IDeckLinkOutput::StartScheduledPlayback method

The **StartScheduledPlayback** method starts scheduled playback. Frames may be pre-rolled by scheduling them before starting playback. **SetScheduledFrameCompletionCallback** may be used to register a callback to be called when each frame is completed.

Playback starts immediately when **StartScheduledPlayback** is called but at a specified "playback start time". Scheduled frames are output as the playback time reaches the time at which the frames were scheduled.

Syntax

```
HRESULT     StartScheduledPlayback
            (BMDTimeValue playbackStartTime, BMDTimeScale timeScale,
            double playbackSpeed);
```

Parameters

Name	Direction	Description
`playbackStartTime`	in	Time at which the playback starts in units of timeScale
`timeScale`	in	Time scale for playbackStartTime and playbackSpeed.
`playbackSpeed`	in	Speed at which to play back : 1.0 is normal playback, -1.0 is reverse playback. Fast or slow forward or reverse playback may also be specified.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.24 IDeckLinkOutput::StopScheduledPlayback method

The **StopScheduledPlayback** method stops scheduled playback immediately or at a specified time. Any frames or audio scheduled after the stop time will be flushed.

Syntax

```
HRESULT     StopScheduledPlayback
            (BMDTimeValue stopPlaybackAtTime,
            BMDTimeValue *actualStopTime, BMDTimeScale timeScale);
```

Parameters

Name	Direction	Description
`stopPlaybackAtTime`	in	Playback time at which to stop in units of timeScale. Specify 0 to stop immediately.
`actualStopTime`	out	Playback time at which playback actually stopped in units of timeScale. Specify NULL to stop immediately
`timeScale`	in	Time scale for stopPlaybackAtTime and actualStopTime. Specify 0 to stop immediately.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.25 IDeckLinkOutput::GetScheduledStreamTime method

The **GetScheduledStreamTime** method returns the elapsed time since scheduled playback began.

Syntax

```
HRESULT     GetScheduledStreamTime
            (BMDTimeScale desiredTimeScale, BMDTimeValue *streamTime,
            double *playbackSpeed);
```

Parameters

Name	Direction	Description
`desiredTimeScale`	in	Time scale for elapsedTimeSinceSchedulerBegan
`streamTime`	out	Frame time
`playbackSpeed`	out	Scheduled playback speed

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Video output is not enabled

2.5.3.26 IDeckLinkOutput::GetReferenceStatus method

The **GetReferenceStatus** method provides the genlock reference status of the DeckLink device.

Syntax

```
HRESULT      GetReferenceStatus (BMDReferenceStatus *referenceStatus)
```

Parameters

Name	Direction	Description
`referenceStatus`	out	A bit-mask of the reference status. See **BMDReferenceStatus** for more details.

Return Values

Value	Description
E_FAIL	Failure
E_POINTER	The parameter is invalid.
S_OK	Success

2.5.3.27 IDeckLinkOutput::GetHardwareReferenceClock method

The **GetHardwareReferenceClock** method returns a clock that is locked to the rate at which the DeckLink hardware is outputting frames. The absolute values returned by this method are meaningless, however the relative differences between subsequent calls can be used to determine elapsed time. This method can be called while video output is enabled (see **IDeckLinkOutput::EnableVideoOutput** for details).

Syntax

```
HRESULT      GetHardwareReferenceClock
             (BMDTimeScale desiredTimeScale,
             BMDTimeValue *hardwareTime, BMDTimeValue *timeInFrame,
             BMDTimeValue *ticksPerFrame);
```

Parameters

Name	Direction	Description
`desiredTimeScale`	in	Desired time scale
`hardwareTime`	out	Hardware reference time (in units of desiredTimeScale)
`timeInFrame`	out	Time in frame (in units of desiredTimeScale)
`ticksPerFrame`	out	Number of ticks for a frame (in units of desiredTimeScale)

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.3.28 IDeckLinkOutput:: GetFrameCompletionReferenceTimestamp method

The **GetFrameCompletionReferenceTimestamp** method is called to determine the time that the frame has been output.

The timestamp is valid if this method is called within the **ScheduledFrameCompleted** callback and if the frame referenced by the Frame pointer has not been re-scheduled.

Syntax

```
HRESULT     GetFrameCompletionReferenceTimestamp
            (IDeckLinkVideoFrame *theFrame,
            BMDTimeScale desiredTimeScale,
            BMDTimeValue *frameCompletionTimestamp)
```

Parameters

Name	Direction	Description
`desiredTimeScale`	in	Desired time scale
`theFrame`	in	The video frame
`desiredTimeScale`	in	Desired timescale
`frameCompletionTimestamp`	out	Timestamp that the frame completed (in units of desiredTimeScale).

Return Values

Value	Description
E_UNEXPECTED	A timestamp for the specified frame is not available.
S_OK	Success

2.5.4 IDeckLinkInput Interface

The **IDeckLinkInput** object interface allows an application to capture a video and audio stream from a DeckLink device.

An **IDeckLinkInput** interface can be obtained from an **IDeckLink** object interface using **QueryInterface**. If QueryInterface for an input interface is called on an output only device, then QueryInterface will fail and return E_NOINTERFACE.

Video capture operates in a push model with each video frame being delivered to an **IDeckLinkInputCallback** object interface. Audio capture is optional and can be handled by using the same callback.

Please note that non-4K DeckLink devices and sub-devices are half-duplex. Therefore either capture or render can be enabled, but not simultaneously.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	An **IDeckLinkInput** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkDisplayModeIterator`	`IID_IDeckLink DisplayModeIterator`	**IDeckLinkInput::GetDisplayModeIterator** returns an **IDeckLinkDisplayModeIterator** object interface
`IDeckLinkInputCallback`	`IID_DeckLinkInputCallback`	An **IDeckLinkInputCallback** object interface may be registered with **IDeckLinkInput::SetCallback**

Public Member Functions	
Method	**Description**
DoesSupportVideoMode	Check whether a given video mode is supported for input
GetDisplayModeIterator	Get an iterator to enumerate the available input display modes
SetScreenPreviewCallback	Register screen preview callback
EnableVideoInput	Configure video input
GetAvailableVideoFrameCount	Query number of available video frames
DisableVideoInput	Disable video input
EnableAudioInput	Configure audio input
DisableAudioInput	Disable audio input
GetBufferedAudioSampleFrameCount	Query audio buffer status – for pull model audio.
StartStreams	Start synchronized capture
StopStreams	Stop synchronized capture
PauseStreams	Pause synchronized capture
FlushStreams	Removes any buffered video and audio frames.
SetCallback	Register input callback
GetHardwareReferenceClock	Get the hardware system clock
SetVideoInputFrameMemoryAllocator	Register custom memory allocator for input video frames

2.5.4.1 IDeckLinkInput::DoesSupportVideoMode method

The **DoesSupportVideoMode** method indicates whether a given display mode is supported on input.

Modes may be supported, unsupported or supported with conversion.

Syntax

```
HRESULT     DoesSupportVideoMode
            (BMDDisplayMode displayMode, BMDPixelFormat pixelFormat,
            BMDVideoInputFlags flags, BMDDisplayModeSupport
            *support, IDeckLinkDisplayMode **resultDisplayMode);
```

Parameters

Name	Direction	Description
`displayMode`	in	Display mode to check
`pixelFormat`	in	Pixel format to check (0 for any)
`flags`	in	Video output flags (see **BMDVideoInputFlags** for details).
`support`	out	Video output mode supported result.
`resultDisplayMode`	out	If this parameter is not NULL, an **IDeckLinkDisplayMode** object representing the given displayMode is returned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.2 IDeckLinkInput::GetDisplayModeIterator method

The **GetDisplayModeIterator** method returns an iterator which enumerates the available display modes.

Syntax

```
HRESULT     GetDisplayModeIterator
            (IDeckLinkDisplayModeIterator **iterator);
```

Parameters

Name	Direction	Description
`iterator`	out	display mode iterator

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.3 IDeckLinkInput::SetScreenPreviewCallback method

The **SetScreenPreviewCallback** method is called to register an instance of an **IDeckLinkScreenPreviewCallback** object. The registered object facilitates the updating of an on-screen preview of a video stream being captured.

Syntax

```
HRESULT     SetScreenPreviewCallback
            (IDeckLinkScreenPreviewCallback *previewCallback)
```

Parameters

Name	Direction	Description
previewCallback	in	The IDeckLinkScreenPreview object to be registered.

Return Values

Value	Description
S_OK	Success

2.5.4.4 IDeckLinkInput::EnableVideoInput method

The **EnableVideoInput** method configures video input and puts the hardware into video capture mode. Video input (and optionally audio input) is started by calling **StartStreams**.

Syntax

```
HRESULT     EnableVideoInput
            (BMDDisplayMode displayMode, BMDPixelFormat pixelFormat,
            BMDVideoInputFlags flags);
```

Parameters

Name	Direction	Description
displayMode	in	Video mode to capture
pixelFormat	in	Pixel format to capture
flags	in	Capture flags

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	Is returned on invalid mode or video flags
E_ACCESSDENIED	Unable to access the hardware or input stream currently active
E_OUTOFMEMORY	Unable to create a new frame

2.5.4.5 IDeckLinkInput::GetAvailableVideoFrameCount method

The **GetAvailableVideoFrameCount** method provides the number of available input frames.

Syntax

```
HRESULT     GetAvailableVideoFrameCount
            uint32_t *availableFrameCount);
```

Parameters

Name	Direction	Description
availableFrameCount	out	Number of available input frames.

Return Values

Value	Description
S_OK	Success

2.5.4.6 IDeckLinkInput::DisableVideoInput method

The **DisableVideoInput** method disables the hardware video capture mode.

Syntax

```
HRESULT     DisableVideoInput ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.7 IDeckLinkInput::EnableAudioInput method

The **EnableAudioInput** method configures audio input and puts the hardware into audio capture mode. Synchronized audio and video input is started by calling **StartStreams**.

Syntax

```
HRESULT       EnableAudioInput
              (BMDAudioSampleRate sampleRate,
              BMDAudioSampleType sampleType,
              uint32_t channelCount);
```

Parameters

Name	Direction	Description
sampleRate	in	Sample rate to capture
sampleType	in	Sample type to capture
channelCount	in	Number of audio channels to capture – only 2, 8 or 16 channel capture is supported.

Return Values

Value	Description
E_FAIL	Failure
E_INVALIDARG	Invalid number of channels requested
S_OK	Success

2.5.4.8 IDeckLinkInput::DisableAudioInput method

The **DisableAudioInput** method disables the hardware audio capture mode.

Syntax

```
HRESULT       DisableAudioInput ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.9 IDeckLinkInput::GetAvailableAudioSampleFrameCount method

The **GetAvailableAudioSampleFrameCount** method returns the number of audio sample frames currently buffered.

Use of this method is only required when using pull model audio – the same audio data is made available to **IDeckLinkInputCallback** and may be ignored.

Syntax

```
HRESULT     GetAvailableAudioSampleFrameCount
            (uint32_t *availableSampleFrameCount);
```

Parameters

Name	Direction	Description
availableSampleFrameCount	out	The number of buffered audio frames currently available.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.10 IDeckLinkInput::SetVideoInputFrameMemoryAllocator method

The **SetVideoInputFrameMemoryAllocator** method sets a custom memory allocator for video frame allocations during capture. Use of a custom memory allocator is optional.

Syntax

```
HRESULT     SetVideoInputFrameMemoryAllocator
            (IDeckLinkMemoryAllocator *theAllocator);
```

Parameters

Name	Direction	Description
theAllocator	in	Allocator object with an **IDeckLinkMemoryAllocator** interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.11 IDeckLinkInput::StartStreams method

The **StartStreams** method starts synchronized video and audio capture as configured with **EnableVideoInput** and optionally **EnableAudioInput**.

Syntax

```
HRESULT       StartStreams ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Input stream is already running.
E_UNEXPECTED	Video and Audio inputs are not enabled.

2.5.4.12 IDeckLinkInput::StopStreams method

The **StopStreams** method stops synchronized video and audio capture.

Syntax

```
HRESULT       StopStreams ();
```

Parameters

none.

Return Values

Value	Description
S_OK	Success
E_ACCESSDENIED	Input stream already stopped.

2.5.4.13 IDeckLinkInput::FlushStreams method

The **FlushStreams** method removes any buffered video and audio frames.

Syntax

```
HRESULT       FlushStreams ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.14 IDeckLinkInput::PauseStreams method

The **PauseStreams** method pauses synchronized video and audio capture. Capture time continues while the streams are paused but no video or audio will be captured. Paused capture may be resumed by calling **PauseStreams** again. Capture may also be resumed by calling **StartStreams** but capture time will be reset.

Syntax

```
HRESULT      PauseStreams ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.15 IDeckLinkInput::SetCallback method

The **SetCallback** method configures a callback which will be called for each captured frame. Synchronized capture is started with **StartStreams**, stopped with **StopStreams** and may be paused with **PauseStreams**.

Syntax

```
HRESULT      SetCallback (IDeckLinkInputCallback *theCallback);
```

Parameters

Name	Direction	Description
`theCallBack`	in	callback object implementing the **IDeckLinkInputCallback** object interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.4.16 IDeckLinkInput::GetHardwareReferenceClock method

The **GetHardwareReferenceClock** method returns a clock that is locked to the system clock. The absolute values returned by this method are meaningless, however the relative differences between subsequent calls can be used to determine elapsed time. This method can be called while video input is enabled (see **IDeckLinkInput::EnableVideoInput** for details).

Syntax

```
HRESULT     GetHardwareReferenceClock
            (BMDTimeScale desiredTimeScale,
            BMDTimeValue *hardwareTime, BMDTimeValue *timeInFrame,
            BMDTimeValue *ticksPerFrame);
```

Parameters

Name	Direction	Description
`desiredTimeScale`	in	Desired time scale
`hardwareTime`	out	Hardware reference time (in units of desiredTimeScale)
`timeInFrame`	out	Time in frame (in units of desiredTimeScale)
`ticksPerFrame`	out	Number of ticks for a frame (in units of desiredTimeScale)

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.5 IDeckLinkVideoFrame Interface

The **IDeckLinkVideoFrame** object interface represents a video frame.

The **GetWidth**, **GetHeight** methods may be used to determine the pixel dimensions of the frame buffer. Pixels on a given row are packed according to the pixel format returned by **GetPixelFormat** - see **BMDPixelFormat** for details. Note that in some formats (HD720 formats, for example), there is padding between rows - always use **GetRowBytes** to account for the row length, including padding.

Developers may sub-class **IDeckLinkVideoFrame** to provide an implementation which fits well with their application's structure.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkMutableVideoFrame`	`IID_IDeckLinkMutableVideoFrame`	**IDeckLinkMutableVideoFrame** subclasses **IDeckLinkVideoFrame**
`IDeckLinkVideoInputFrame`	`IID_IDeckLinkVideoInputFrame`	**IDeckLinkVideoInputFrame** subclasses **IDeckLinkVideoFrame**
`IDeckLinkVideoFrame AncillaryPackets`	`IID_IDeckLinkVideoFrame AncillaryPackets`	An **IDeckLinkVideoFrameAncillaryPackets** object interface may be obtained from **IDeckLinkVideoFrame** using **QueryInterface**

Public Member Functions	
Method	**Description**
GetWidth	Get video frame width in pixels
GetHeight	Get video frame height in pixels
GetRowBytes	Get bytes per row for video frame
GetPixelFormat	Get pixel format for video frame
GetFlags	Get frame flags
GetBytes	Get pointer to frame data
GetTimecode	Gets timecode information
GetAncillaryData	Gets ancillary data

2.5.5.1 IDeckLinkVideoFrame::GetWidth method

The **GetWidth** method returns the width of a video frame.

Syntax

```
long        GetWidth ();
```

Return Values

Value	Description
Width	Video frame width in pixels

2.5.5.2 IDeckLinkVideoFrame::GetHeight method

The **GetHeight** method returns the height of a video frame.

Syntax

```
long        GetHeight ();
```

Return Values

Value	Description
Height	Video frame height in pixels

2.5.5.3 IDeckLinkVideoFrame::GetRowBytes method

The **GetRowBytes** method returns the number of bytes per row of a video frame.

Syntax

```
long        GetRowBytes ();
```

Return Values

Value	Description
BytesCount	Number of bytes per row of video frame

2.5.5.4 IDeckLinkVideoFrame::GetPixelFormat method

The **GetPixelFormat** method returns the pixel format of a video frame.

Syntax

```
BMDPixelFormat      GetPixelFormat ();
```

Return Values

Value	Description
PixelFormat	Pixel format of video frame (**BMDPixelFormat**)

2.5.5.5 IDeckLinkVideoFrame::GetFlags method

The **GetFlags** method returns status flags associated with a video frame.

Syntax

```
BMDFrameFlags       GetFlags ();
```

Return Values

Value	Description
FrameFlags	Video frame flags (**BMDFrameFlags**)

2.5.5.6 IDeckLinkVideoFrame::GetBytes method

The **GetBytes** method allows direct access to the data buffer of a video frame.

Syntax

```
HRESULT      GetBytes (void **buffer);
```

Parameters

Name	Direction	Description
`buffer`	out	Pointer to raw frame buffer – only valid while object remains valid.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.5.7 IDeckLinkVideoFrame::GetTimecode method

The **GetTimecode** method returns the value specified in the ancillary data for the specified timecode type. If the specified timecode type is not found or is invalid, **GetTimecode** returns **S_FALSE**.

Syntax

```
HRESULT      GetTimecode (BMDTimecodeFormat format,
             IDeckLinkTimecode **timecode)
```

Parameters

Name	Direction	Description
`format`	in	**BMDTimecodeFormat** to query
`timecode`	out	New **IDeckLinkTimecode** object interface containing the requested timecode or NULL if requested timecode is not available.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	An invalid or unsupported timecode format was requested.
S_FALSE	The requested timecode format was not present or valid in the ancillary data.

2.5.5.8 IDeckLinkVideoFrame::GetAncillaryData method

The **GetAncillaryData** method returns a pointer to a video frame's ancillary data.

Syntax

```
HRESULT      GetAncillaryData
             (IDeckLinkVideoFrameAncillary **ancillary)
```

Parameters

Name	Direction	Description
`ancillary`	out	Pointer to a new **IDeckLinkVideoFrameAncillary** object. This object must be released by the caller when no longer required.

Return Values

Value	Description
S_OK	Success
S_FALSE	No ancillary data present.

2.5.6 IDeckLinkVideoOutputCallback Interface

The **IDeckLinkVideoOutputCallback** object interface is a callback class which is called for each frame as its processing is completed by the DeckLink device.

An object with an **IDeckLinkVideoOutputCallback** object interface may be registered as a callback with the **IDeckLinkOutput** object interface.

IDeckLinkVideoOutputCallback should be used to monitor frame output statuses and queue a replacement frame to maintain streaming playback. If the application is managing its own frame buffers, they should be disposed or reused inside the **ScheduledFrameCompleted** callback.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkVideoOutputCallback** object interface may be registered with **IDeckLinkOutput::SetScheduledFrameCompletionCallback**

Public Member Functions	
Method	**Description**
ScheduledFrameCompleted	Called when playback of a scheduled frame is completed
ScheduledPlaybackHasStopped	Called when playback has stopped.

2.5.6.1 IDeckLinkVideoOutputCallback:: ScheduledFrameCompleted method

The **ScheduledFrameCompleted** method is called when a scheduled video frame playback is completed. This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

The **IDeckLinkVideoOutputCallback** methods are called on a dedicated callback thread. To prevent video frames from being either dropped or delayed, ensure that any application processing on the callback thread takes less time than a frame time. If the application processing time is greater than a frame time, multiple threads should be used.

Syntax

```
HRESULT     ScheduledFrameCompleted
            (IDeckLinkVideoFrame* completedFrame,
            BMDOutputFrameCompletionResult result);
```

Parameters

Name	Direction	Description
`completedFrame`	in	Completed frame
`result`	in	Frame completion result – see **BMDOutputFrameCompletionResult** for details.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.6.2 IDeckLinkVideoOutputCallback:: ScheduledPlaybackHasStopped method

The **ScheduledPlaybackHasStopped** method is called when a scheduled playback has stopped.

Syntax

```
HRESULT     ScheduledPlaybackHasStopped(void)
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.7 IDeckLinkMutableVideoFrame Interface

The **IDeckLinkMutableVideoFrame** object interface represents a video frame created for output. Methods are provided to attach ancillary data and set timecodes within the frame.

IDeckLinkMutableVideoFrame is a subclass of **IDeckLinkVideoFrame** and inherits all its methods. It is created by the **IDeckLinkOutput::CreateVideoFrame** method.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkVideoFrame`	`IID_IDeckLinkVideoFrame`	**IDeckLinkMutableVideoFrame** subclasses **IDeckLinkVideoFrame**

Public Member Functions	
Method	**Description**
SetFlags	Set flags applicable to a video frame
SetTimecode	Set timecode
SetTimecodeFromComponents	Set components of specified timecode type
SetAncillaryData	Set frame ancillary data
SetTimecodeUserBits	Set the timecode user bits

2.5.7.1 IDeckLinkMutableVideoFrame::SetFlags method

The **SetFlags** method sets output flags associated with a video frame.

Syntax

```
HRESULT      SetFlags (BMDFrameFlags newFlags);
```

Parameters

Name	Direction	Description
`newFlags`	in	**BMDFrameFlags** to set - see **BMDFrameFlags** for details.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.7.2 IDeckLinkMutableVideoFrame::SetTimecode method

The **SetTimecode** method sets the specified timecode type for the frame.

Syntax

```
HRESULT     SetTimecode (BMDTimecodeFormat format,
            IDeckLinkTimecode* timecode);
```

Parameters

Name	Direction	Description
format	in	**BMDTimecodeFormat** to update
timecode	in	**IDeckLinkTimecode** object interface containing timecode to copy.

Return Values

Value	Description
E_UNEXPECTED	Unexpected timecode. Ensure that VITC1 has been set.
S_OK	Success

2.5.7.3 IDeckLinkMutableVideoFrame:: SetTimecodeFromComponents method

The **SetTimecodeFromComponents** method sets the components of the specified timecode type for the frame.

Syntax

```
HRESULT     SetTimecodeFromComponents
            (BMDTimecodeFormat format, uint8_t hours,
            uint8_t minutes, uint8_t seconds, uint8_t frames,
            BMDTimecodeFlags flags);
```

Parameters

Name	Direction	Description
format	in	**BMDTimecodeFormat** to update
hours	in	Value of hours component of timecode
minutes	in	Value of minutes component of timecode
seconds	in	Value of seconds component of timecode
frames	in	Value of frames component of timecode
flags	in	Timecode flags (see **BMDTimecodeFlags** for details)

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.7.4 IDeckLinkMutableVideoFrame::SetAncillaryData method

The **SetAncillaryData** method sets frame ancillary data. An **IDeckLinkVideoFrameAncillary** may be created using the **IDeckLinkOutput::CreateAncillaryData** method.

Syntax

```
HRESULT     SetAncillaryData
            (IDeckLinkVideoFrameAncillary* ancillary);
```

Parameters

Name	Direction	Description
ancillary	in	**IDeckLinkVideoFrameAncillary** data to output with the frame.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.7.5 IDeckLinkMutableVideoFrame::SetTimecodeUserBits method

The **SetTimecodeUserBits** method sets the timecode user bits.

Syntax

```
HRESULT     SetTimecodeUserBits
            (BMDTimecodeFormat format, BMDTimecodeUserBits userBits)
```

Parameters

Name	Direction	Description
format	in	The format of the timecode.
userBits	in	The user bits to set.

Return Values

Value	Description
E_NOTIMPL	Not implemented
E_INVALIDARG	The format parameter is invalid.
E_UNEXPECTED	Timecode object is not present. See: **IDeckLinkMutableVideoFrame::SetTimecode**

2.5.8 IDeckLinkVideoFrame3DExtensions Interface

The **IDeckLinkVideoFrame3DExtensions** object interface allows linking of video frames in left eye / right eye pairs, to support 3D capture and playback.

This interface is applicable only to DeckLink devices which support 3D features, such the DeckLink 4K Extreme. All frames belonging to a 3D stream carry an **IDeckLinkVideoFrame3DExtensions** object, which indicates whether this frame is a left- or right-eye frame and allows access to the right eye frame if this frame is a left eye frame.

When capturing in a 3D video mode, an **IDeckLinkVideoFrame3DExtensions** object can be obtained by calling **IDeckLinkVIdeoFrame::QueryInterface** on frames returned by the API.

When outputting in a 3D video mode, your application must provide video frame objects which implement the **IDeckLinkVideoFrame** interface and return a valid **IDeckLinkVideoFrame3DExtensions** object. See section 2.3.3.

An **IDeckLinkVideoFrame3DExtensions** object can be obtained:

- From **IDeckLinkVideoInputFrame** using **QueryInterface**, if capturing in 3D mode has been enabled (see **IDeckLinkInput::Enable** and **bmdVideoInputDualStream3D** for details) or by subclassing **IDeckLinkVideoInputFrame**. By subclassing **IDeckLinkVideoFrame3DExtensions**.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkVideoFrame`	`IID_IDeckLinkVideoFrame`	When capturing in a 3D mode, an **IDeckLinkVideoFrame3DExtensions** may be obtained from **IDeckLinkVideoFrame** using **QueryInterface**

Public Member Functions	
Method	**Description**
Get3DPackingFormat	The indication of whether the frame represents the left or the right eye.
GetFrameForRightEye	Get the right eye frame of a 3D pair.

2.5.8.1 IDeckLinkVideoFrame3DExtensions:: Get3DPackingFormat method

The **Get3DPackingFormat** method indicates whether the video frame belongs to the left eye or right eye stream.

Syntax

```
BMDVideo3DPackingFormat   Get3DPackingFormat (void)
```

Return Values

Value	Description
Packing format	Either **bmdVideo3DPackingRightOnly** or **bmdVideo3DPackingLeftOnly**. See **BMDVideo3DPackingFormat** for more details.

2.5.8.2 IDeckLinkVideoFrame3DExtensions:: GetFrameForRightEye method

The **GetFrameForRightEye** method accesses the right eye frame of a 3D pair.

Syntax

```
HRESULT      GetFrameForRightEye
             (IDeckLinkVideoFrame* *rightEyeFrame)
```

Parameters

Name	Direction	Description
`rightEyeFrame`	out	The right eye frame. This object must be released by the caller when no longer required.

Return Values

Value	Description
E_INVALIDARG	The parameter is invalid.
S_FALSE	This frame is the right eye frame.
S_OK	Success

2.5.9 IDeckLinkAudioOutputCallback Interface

The **IDeckLinkAudioOutputCallback** object interface is a callback class called regularly during playback to allow the application to check for the amount of audio currently buffered and buffer more audio if required.

An **IDeckLinkAudioOutputCallback** object interface may be registered with **IDeckLinkOutput::SetAudioCallback**.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkAudioOutputCallback** object interface may be registered with **IDeckLinkOutput::SetAudioCallback**

Public Member Functions	
Method	**Description**
RenderAudioSamples	Called to allow buffering of more audio samples if required

2.5.9.1 IDeckLinkAudioOutputCallback:: RenderAudioSamples method

The **RenderAudioSamples** method is called at a rate of 50Hz during playback. When audio preroll is enabled with a call to **IDeckLinkOutput::BeginAudioPreroll**, the **RenderAudioSamples** method is called continuously until either **IDeckLinkOutput::EndAudioPreroll** or **IDeckLinkOutput::StartScheduledPlayback** is called.

During preroll (preroll is TRUE) call **IDeckLinkOutput::ScheduleAudioSamples** to schedule sufficient audio samples for the number of video frames that have scheduled.

During playback (preroll is FALSE) check the count of buffered audio samples with **IDeckLinkOutput::GetBufferedAudioSampleFrameCount** and when required, schedule more audio samples with **IDeckLinkOutput::ScheduleAudioSamples**.

Syntax

```
HRESULT      RenderAudioSamples (boolean preroll);
```

Parameters

Name	Direction	Description
`preroll`	in	Flag specifying whether driver is currently pre-rolling (TRUE) or playing (FALSE).

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.10 IDeckLinkInputCallback Interface

The **IDeckLinkInputCallback** object interface is a callback class which is called for each captured frame.

An object with an **IDeckLinkInputCallback** interface may be registered as a callback with the **IDeckLinkInput** object interface.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkInput`	`IID_IDeckLinkInput`	An **IDeckLinkInputCallback** object interface may be registered with **IDeckLinkInput::SetCallback**
`IDeckLinkVideoInputFrame`	`IID_DeckLinkVideoInputFrame`	**An IDeckLinkVideoInputFrame** object interface is passed to **IDeckLinkInputCallback::VideoInputFrameArrived**
`IDeckLinkAudioInputPacket`	`IID_DeckLinkAudioInputPacket`	An **IDeckLinkAudioInputPacket** object interface is passed to **IDeckLinkInputCallback::VideoInputFrameArrived**

Public Member Functions	
Method	**Description**
VideoInputFrameArrived	Called when new video data is available
VideoInputFormatChanged	Called when a video input format change is detected

2.5.10.1 IDeckLinkInputCallback::VideoInputFrameArrived method

The **VideoInputFrameArrived** method is called when a video input frame or an audio input packet has arrived. This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT     VideoInputFrameArrived
            (IDeckLinkVideoInputFrame *videoFrame,
            IDeckLinkAudioInputPacket *audioPacket);
```

Parameters

Name	Direction	Description
`videoFrame`	in	The video frame that has arrived. The video frame is only valid for the duration of the callback. To hold on to the video frame beyond the callback call **AddRef**, and to release the video frame when it is no longer required call **Release**. The video frame will be NULL under the following circumstances: - On Intensity Pro with progressive NTSC only, every video frame will have two audio packets. - With 3:2 pulldown there are five audio packets for each four video frames. - If video processing is not fast enough, audio will still be delivered.
`audioPacket`	in	New audio packet-only valid if audio capture has been enabled with **IDeckLinkInput::EnableAudioInput** The audio packet will be NULL under the following circumstances: - Audio input is not enabled. - If video processing is sufficiently delayed old video may be received with no audio.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.10.2 IDeckLinkInputCallback::VideoInputFormatChanged method

The **VideoInputFormatChanged** method is called when a video input format change has been detected by the hardware.

To enable this feature, the **bmdVideoInputEnableFormatDetection** flag must set when calling **IDeckLinkInput::EnableVideoInput()**.

Note: The video format change detection feature is not currently supported on all hardware. Check the **BMDDeckLinkSupportsInputFormatDetection** attribute to determine if this feature is supported for a given device and driver (see **IDeckLinkAttributes** Interface for details).

Syntax

```
HRESULT     VideoInputFormatChanged
            (BMDVideoInputFormatChangedEvents notificationEvents,
            IDeckLinkDisplayMode *newDisplayMode,
            BMDDetectedVideoInputFormatFlags detectedSignalFlags);
```

Parameters

Name	Direction	Description
notificationEvents	in	The notification events - enable input detection
newDisplayMode	in	The new display mode.
detectedSignalFlags	in	The detected signal flags

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.11 IDeckLinkVideoInputFrame Interface

The **IDeckLinkVideoInputFrame** object interface represents a video frame which has been captured by an **IDeckLinkInput** object interface. **IDeckLinkVideoInputFrame** is a subclass of **IDeckLinkVideoFrame** and inherits all its methods.

Objects with an **IDeckLinkVideoInputFrame** interface are passed to the **IDeckLinkInputCallback::VideoInputFrameArrived** callback.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkInput`	`IID_IDeckLinkInput`	New input frames are returned to **IDeckLinkInputCallback::VideoInputFrameArrived** by the **IDeckLinkInput** interface
`IDeckLinkVideoFrame`	`IID_IDeckLinkVideoFrame`	**IDeckLinkVideoInputFrame** subclasses **IDeckLinkVideoFrame**

Public Member Functions	
Method	**Description**
GetStreamTime	Get video frame timing information
GetHardwareReferenceTimestamp	Get hardware reference timestamp

2.5.11.1 IDeckLinkVideoInputFrame::GetStreamTime method

The **GetStreamTime** method returns the time and duration of a captured video frame for a given timescale.

Syntax

```
HRESULT     GetStreamTime
            (BMDTimeValue *frameTime, BMDTimeValue *frameDuration,
            BMDTimeScale timeScale);
```

Parameters

Name	Direction	Description
`frameTime`	out	Frame time (in units of timeScale)
`frameDuration`	out	Frame duration (in units of timeScale)
`timeScale`	in	Time scale for output parameters

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.11.2 IDeckLinkVideoInputFrame:: GetHardwareReferenceTimestamp method

The **GetHardwareReferenceTimestamp** method returns frame time and frame duration for a given timescale.

Syntax

```
HRESULT     GetHardwareReferenceTimestamp
            (BMDTimeScale timeScale, BMDTimeValue *frameTime,
            BMDTimeValue *frameDuration);
```

Parameters

Name	Direction	Description
`timeScale`	in	The time scale - see **BMDTimeScale** for details.
`frameTime`	out	The frame time - see **BMDTimeValue** for details.
`frameDuration`	out	The frame duration - see **BMDTimeValue** for details.

Return Values

Value	Description
E_INVALIDARG	Timescale is not set
S_OK	Success

2.5.12 IDeckLinkAudioInputPacket Interface

The **IDeckLinkAudioInputPacket** object interface represents a packet of audio which has been captured by an **IDeckLinkInput** object interface.

Objects with an **IDeckLinkAudioInputPacket** object interface are passed to the **IDeckLinkInputCallback::VideoInputFrameArrived** callback.

Audio channel samples are interleaved into a sample frame and sample frames are contiguous.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkInputCallback`	`IID_IDeckLinkInputCallback`	New audio packets are returned to the **IDeckLinkInputCallback::VideoInputFrameArrived** callback

Public Member Functions	
Method	**Description**
GetSampleFrameCount	Get number of sample frames in packet
GetBytes	Get pointer to raw audio frame sequence
GetPacketTime	Get corresponding video timestamp

2.5.12.1 IDeckLinkAudioInputPacket::GetSampleFrameCount method

The **GetSampleFrameCount** method returns the number of sample frames in the packet.

Syntax

```
long        GetSampleFrameCount ();
```

Return Values

Value	Description
Count	Audio packet size in sample frames

2.5.12.2 IDeckLinkAudioInputPacket::GetBytes method

The **GetBytes** method returns a pointer to the data buffer of the audio packet.

Syntax

```
HRESULT     GetBytes (void **buffer);
```

Parameters

Name	Direction	Description
`buffer`	out	pointer to audio data – only valid while object remains valid

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.12.3 IDeckLinkAudioInputPacket::GetPacketTime method

The **GetPacketTime** method returns the time stamp of the video frame corresponding to the specified audio packet.

Syntax

```
HRESULT      GetPacketTime
             (BMDTimeValue *packetTime, BMDTimeScale timeScale);
```

Parameters

Name	Direction	Description
`packetTime`	out	Video frame time corresponding to audio packet in timeScale units
`timeScale`	in	Time scale for time stamp to be returned

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.13 IDeckLinkDisplayModeIterator Interface

The **IDeckLinkDisplayModeIterator** object interface is used to enumerate the available display modes for a DeckLink device.

An **IDeckLinkDisplayModeIterator** object interface may be obtained from an **IDeckLinkInput** or **IDeckLinkOutput** object interface using the **GetDisplayModeIterator** method.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkInput`	`IID_IDeckLinkInput`	**IDeckLinkInput::GetDisplayModeIterator** returns an **IDeckLinkDisplayModeIterator** object interface
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	**IDeckLinkOutput::GetDisplayModeIterator** returns an **IDeckLinkDisplayModeIterator** object interface
`IDeckLinkDisplayMode`	`IID_IDeckLinkDisplayMode`	**IDeckLinkDisplayModeIterator::Next** returns an **IDeckLinkDisplayMode** object interface for each available display mode

Public Member Functions	
Method	**Description**
Next	Returns a pointer to an **IDeckLinkDisplayMode** interface for an available display mode

2.5.13.1 IDeckLinkDisplayModeIterator::Next method

The **Next** method returns the next available **IDeckLinkDisplayMode** interface.

Syntax

```
HRESULT      Next (IDeckLinkDisplayMode **displayMode);
```

Parameters

Name	Direction	Description
`displayMode`	out	**IDeckLinkDisplayMode** object interface or NULL when no more display modes are available.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.14 IDeckLinkDisplayMode Interface

The **IDeckLinkDisplayMode** object interface represents a supported display mode.

The **IDeckLinkDisplayModeIterator** object interface enumerates supported display modes, returning **IDeckLinkDisplayMode** object interfaces.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkDisplayMode Iterator`	`IID_IDeckLinkDisplayMode Iterator`	**IDeckLinkDisplayModeIterator::Next** returns an **IDeckLinkDisplayMode** object interface for each available display mode

Public Member Functions	
Method	**Description**
GetWidth	Get video frame width in pixels
GetHeight	Get video frame height in pixels
GetName	Get descriptive text
GetDisplayMode	Get corresponding **BMDDisplayMode**
GetFrameRate	Get the frame rate of the display mode
GetFieldDominance	Gets the field dominance of the frame
GetFlags	Returns flags associated with display modes (see **BMDDisplaymodeFlags** for more details).

2.5.14.1 IDeckLinkDisplayMode::GetWidth method

The **GetWidth** method returns the width of a video frame in the display mode.

Syntax

```
long        GetWidth ();
```

Return Values

Value	Description
Width	Video frame width in pixels

2.5.14.2 IDeckLinkDisplayMode::GetHeight method

The **GetHeight** method returns the height of a video frame in the display mode.

Syntax

```
long        GetHeight ();
```

Return Values

Value	Description
Height	Video frame height in pixels

2.5.14.3 IDeckLinkDisplayMode::GetName method

The **GetName** method returns a string describing the display mode.

Syntax

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

Parameters

Name	Direction	Description
`name`	out	Descriptive string This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.14.4 IDeckLinkDisplayMode::GetDisplayMode method

The **GetDisplayMode** method returns the corresponding **BMDDisplayMode** for the selected display mode.

Syntax

```
BMDDisplayMode      GetDisplayMode ();
```

Return Values

Value	Description
mode	**BMDDisplayMode** corresponding to the display mode

2.5.14.5 IDeckLinkDisplayMode::GetFrameRate method

The **GetFrameRate** method returns the frame rate of the display mode. The frame rate is represented as the two integer components of a rational number for accuracy. The actual frame rate can be calculated by timeScale / timeValue.

Syntax

```
HRESULT      GetFrameRate
             (BMDTimeValue *timeValue, BMDTimeScale *timeScale);
```

Parameters

Name	Direction	Description
`timeValue`	out	Frame rate value
`timeScale`	out	Frame rate scale

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.14.6 IDeckLinkDisplayMode::GetFieldDominance method

The **GetFieldDominance** method gets the field dominance of the frame.

Syntax

```
BMDFieldDominance       GetFieldDominance ();
```

Return Values

Value	Description
FieldDominance	The field dominance - see **BMDFieldDominance** for details.

2.5.14.7 IDeckLinkDisplayMode::GetFlags method

The **GetFlags** method returns flags associated with display modes.

Syntax

```
BMDDisplayModeFlags     GetFlags ();
```

Return Values

Value	Description
Flags	The display mode flags - see **BMDDisplaymodeFlags** for details.

2.5.15 IDeckLinkConfiguration Interface

The **IDeckLinkConfiguration** object interface allows querying and modification of DeckLink configuration parameters.

An **IDeckLinkConfiguration** object interface can be obtained from the **IDeckLink** interface using **QueryInterface**.

The configuration settings are globally visible (not limited to the current process). Changes will persist until the **IDeckLinkConfiguration** object is released, unless **WriteConfigurationToPreferences** is called. In which case, the changes will be made permanent and will persist across restarts.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	DeckLink device interface

Public Member Functions	
Method	**Description**
SetFlag	Sets a boolean value into the configuration setting associated with the given **BMDDeckLinkConfigurationID**.
GetFlag	Gets the current boolean value of a setting associated with the given **BMDDeckLinkConfigurationID**.
SetInt	Sets the current int64_t value into the configuration setting associated with the given **BMDDeckLinkConfigurationID**.
GetInt	Gets the current int64_t value of a setting associated with the given **BMDDeckLinkConfigurationID**.
SetFloat	Sets the current double value into the configuration setting associated with the given **BMDDeckLinkConfigurationID**.
GetFloat	Gets the current double value of a setting associated with the given **BMDDeckLinkConfigurationID**.
SetString	Sets the current string value into the configuration setting with the given **BMDDeckLinkConfigurationID**.
GetString	Gets the current string value of a setting associated with the given **BMDDeckLinkConfigurationID**.
WriteConfigurationToPreferences	Saves the current settings to system preferences so that they will persist across system restarts.

2.5.15.1 IDeckLinkConfiguration::SetFlag method

The **SetFlag** method sets a boolean value into the configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      SetFlag (BMDDeckLinkConfigurationID cfgID,
                      boolean value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The boolean value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.2 IDeckLinkConfiguration::GetFlag method

The **GetFlag** method gets the current boolean value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      GetFlag (BMDDeckLinkConfigurationID cfgID,
                      boolean *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The boolean value that is set in the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.3 IDeckLinkConfiguration::SetInt method

The **SetInt** method sets the current int64_t value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      SetInt (BMDDeckLinkConfigurationID cfgID,
                     int64_t value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The integer value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no integer type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.4 IDeckLinkConfiguration::GetInt method

The **GetInt** method gets the current int64_t value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      GetInt (BMDDeckLinkConfigurationID cfgID,
                     int64_t *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The integer value that is set in the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no integer type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.5 IDeckLinkConfiguration::SetFloat method

The **SetFloat** method sets the current double value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      SetFloat (BMDDeckLinkConfigurationID cfgID,
                       double value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The double value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.6 IDeckLinkConfiguration::GetFloat method

The **GetFloat** method gets the current double value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      GetFloat (BMDDeckLinkConfigurationID cfgID,
                       double *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The double value that is set in the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.7 IDeckLinkConfiguration::SetString method

The **SetString** method sets the current string value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      SetString (BMDDeckLinkConfigurationID cfgID,
                        string value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The string to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.8 IDeckLinkConfiguration::GetString method

The **GetString** method gets the current string value of a configuration setting associated with the given **BMDDeckLinkConfigurationID**.

Syntax

```
HRESULT      GetString (BMDDeckLinkConfigurationID cfgID,
                        string *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The string set in the selected configuration setting. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type configuration setting for this operation corresponding to the given **BMDDeckLinkConfigurationID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.15.9 IDeckLinkConfiguration:: WriteConfigurationToPreferences method

The **WriteConfigurationToPreferences** method saves the current settings to system preferences so they will persist across system restarts.
This method requires administrative privileges. Configuration settings changed through this interface will be reverted when the interface is released unless this method is called.

Syntax

```
HRESULT      WriteConfigurationToPreferences ();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Insufficient privileges to write to system preferences.

2.5.16 IDeckLinkAPIInformation Interface

The **IDeckLinkAPIInformation** object interface provides global API information. A reference to an **IDeckLinkAPIInformation** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateDeckLinkAPIInformationInstance** on other platforms.

Public Member Functions	
Method	**Description**
GetFlag	Gets a boolean flag associated with specified **BMDDeckLinkAPIInformationID**
GetInt	Gets an int64_t associated with specified **BMDDeckLinkAPIInformationID**
GetFloat	Gets a float associated with specified **BMDDeckLinkAPIInformationID**
GetString	Gets a string associated with specified **BMDDeckLinkAPIInformationID**

2.5.16.1 IDeckLinkAPIInformation::GetFlag method

The **GetFlag** method gets a boolean flag associated with a given **BMDDeckLinkAPIInformationID**.

Syntax

```
HRESULT      GetFlag (BMDDeckLinkAPIInformationID cfgID,
                      bool *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAPIInformationID** to get flag value.
`value`	out	Value of flag corresponding to cfgID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type attribute corresponding to cfgID.

2.5.16.2 IDeckLinkAPIInformation::GetInt method

The **GetInt** method gets an int64_t value associated with a given **BMDDeckLinkAPIInformationID.**

Syntax

```
HRESULT      GetInt (BMDDeckLinkAPIInformationID cfgID,
                     int64_t *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAPIInformationID** to get int value.
`value`	out	Value of int corresponding to cfgID.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	There is no int type attribute corresponding to cfgID.

2.5.16.3 IDeckLinkAPIInformation::GetFloat method

The **GetFloat** method gets a float value associated with a given **BMDDeckLinkAPIInformationID.**

Syntax

```
HRESULT      GetFloat (BMDDeckLinkAPIInformationID cfgID,
                       double *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAPIInformationID** to get float value.
`value`	out	Value of float corresponding to cfgID.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	There is no float type attribute corresponding to cfgID.

2.5.16.4 IDeckLinkAPIInformation::GetString method

The **GetString** method gets a string value associated with a given **BMDDeckLinkAPIInformationID**.

Syntax

```
HRESULT      GetString (BMDDeckLinkAPIInformationID cfgID,
                        String *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAPIInformationID** to get string value.
`value`	out	Value of string corresponding to cfgID.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	There is no string type attribute corresponding to cfgID.
E_OUTOFMEMORY	Unable to allocate memory for string

2.5.17 IDeckLinkAttributes Interface

The **IDeckLinkAttributes** object interface provides details about the capabilities of a DeckLink card. The detail types that are available for various capabilities are: flag, int, float, and string. The DeckLink Attribute ID section lists the hardware capabilities and associated attributes identifiers that can be queried using this object interface. An **IDeckLinkAttributes** object interface can be obtained from the **IDeckLink** interface using **QueryInterface**.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	DeckLink device interface

Public Member Functions	
Method	**Description**
GetFlag	Gets a boolean flag corresponding to a **BMDDeckLinkAttributeID**
GetInt	Gets an int64_t corresponding to a **BMDDeckLinkAttributeID**
GetFloat	Gets a float corresponding to a **BMDDeckLinkAttributeID**
GetString	Gets a string corresponding to a **BMDDeckLinkAttributeID**

2.5.17.1 IDeckLinkAttributes::GetFlag method

The **GetFlag** method gets a boolean flag associated with a given **BMDDeckLinkAttributeID**. (See **BMDDeckLinkAttributeID** for a list of attribute IDs)

Syntax

```
HRESULT      GetFlag (BMDDeckLinkAttributeID cfgID, boolean *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAttributeID** to get flag value.
`value`	out	The value corresponding to cfgID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type attribute corresponding to cfgID.

2.5.17.2 IDeckLinkAttributes::GetInt method

The **GetInt** method gets an **int64_t** value associated with a given **BMDDeckLinkAttributeID**.

Syntax

```
HRESULT      GetInt (BMDDeckLinkAttributeID cfgID, int64_t *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAttributeID** to get int value.
`value`	out	The value corresponding to cfgID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no int type attribute corresponding to cfgID.

2.5.17.3 IDeckLinkAttributes::GetFloat method

The **GetFloat** method gets a float value associated with a given **BMDDeckLinkAttributeID**.

Syntax

```
HRESULT      GetFloat (BMDDeckLinkAttributeID cfgID, double *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAttributeID** to get float value.
`value`	out	The value corresponding to cfgID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type attribute corresponding to cfgID.

2.5.17.4 IDeckLinkAttributes::GetString method

The **GetString** method gets a string value associated with a given **BMDDeckLinkAttributeID**.

Syntax

```
HRESULT      GetString (BMDDeckLinkAttributeID cfgID, string *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDDeckLinkAttributeID** to get string value.
`value`	out	The value corresponding to cfgID. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type attribute corresponding to cfgID.

2.5.18 IDeckLinkMemoryAllocator Interface

The **IDeckLinkMemoryAllocator** object interface is a callback class used to provide control over the memory intensive video frame allocations required during playback and capture. An object with the **IDeckLinkMemoryAllocator** object interface may be registered as a callback with the **IDeckLinkOutput** or **IDeckLinkInput** interfaces.

During playback or capture, calls will be made to this interface object to manage memory buffers for storing video frame data. Memory buffers may be allocated and released more frequently than once per video frame played back or captured, such as when video format conversion is performed.

Implementation of this interface is optional - if this callback is not registered, a default allocator will be used.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkMemoryAllocator** object interface may be registered with **IDeckLinkOutput::SetVideoOutputFrameMemoryAllocator**
`IDeckLinkInput`	`IID_IDeckLinkInput`	An **IDeckLinkMemoryAllocator** object interface may be registered with **IDeckLinkInput::SetVideoInputFrameMemoryAllocator**

Public Member Functions	
Method	**Description**
AllocateBuffer	Called to allocate memory for a frame
ReleaseBuffer	Called to release a previously allocated frame
Commit	Called to notify the allocator that frame buffers will be required
Decommit	Called to notify the allocator that frame buffers will no longer be required (until next call to Commit).

2.5.18.1 IDeckLinkMemoryAllocator::AllocateBuffer method

The **AllocateBuffer** method is called by the owner interface to allocate a buffer for a video frame. This method is abstract in the base interface and must be implemented by the application developer.

Syntax

```
HRESULT      AllocateBuffer (unsigned long bufferSize,
                             void **allocatedBuffer);
```

Parameters

Name	Direction	Description
`bufferSize`	in	Size of the memory to be allocated for a new video frame
`allocatedBuffer`	out	Address of newly allocated buffer ***Note:*** Returned address for buffer must be aligned on a 16-byte boundary.

Return Values

Value	Description
S_OK	Success
E_OUTOFMEMORY	There is insufficient memory to allocate a buffer of the requested size.

2.5.18.2 IDeckLinkMemoryAllocator::ReleaseBuffer method

The **ReleaseBuffer** method is called by the owner interface to release previously allocated memory. This method is abstract in the base interface and must be implemented by the application developer.

Syntax

```
HRESULT      ReleaseBuffer (void *buffer);
```

Parameters

Name	Direction	Description
`buffer`	in	Pointer to the buffer to be released

Return Values

Value	Description
S_OK	Success

2.5.18.3 IDeckLinkMemoryAllocator::Commit method

The **Commit** method is called by the owner interface to notify the allocator that frame buffers will be required. The allocator should allocate any structures required for memory pool management in this callback. This method is abstract in the base interface and must be implemented by the application developer.

Syntax

```
HRESULT       Commit ();
```

Parameters

none.

Return Values

Value	Description
S_OK	Success
E_OUTOFMEMORY	There is insufficient memory to allocate a buffer of the requested size.

2.5.18.4 IDeckLinkMemoryAllocator::Decommit method

The **Decommit** method is called by the owner interface to notify the allocator that frame buffers will no longer be required. The allocator should de-allocate any structures required for memory pool management in this callback. The owner interface will call the Commit method again before allocating more frames. This method is abstract in the base interface and must be implemented by the application developer.

Syntax

```
HRESULT       Decommit ();
```

Parameters

none.

Return Values

Value	Description
S_OK	Success

2.5.19 IDeckLinkKeyer Interface

The **IDeckLinkKeyer** object interface allows configuration of the keying functionality available on most DeckLink cards. An **IDeckLinkKeyer** object interface can be obtained from the **IDeckLink** interface using **QueryInterface**.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	DeckLink device interface

Public Member Functions	
Method	**Description**
Enable	Turn on keyer.
SetLevel	Set the level that the image is blended into the frame.
RampUp	Progressively blends in an image over a given number of frames
RampDown	Progressively blends out an image over a given number of frames
Disable	Turn off keyer

2.5.19.1 IDeckLinkKeyer::Enable method

The **Enable** method turns on the keyer functionality. The **IDeckLinkAttributes** interface can be used to determine if hardware supports the keyer functionality. If external keying is selected, the mask is output on CH A and the key on CH B. The following table lists the hardware that support various keyer capabilities. Currently capture of mask/key on dual channel inputs is not supported.

The following table displays hardware which supports the keyer functionality.

Device	Internal	External	SD	HD to p30	HD to p60	UHD to p30	UHD to p60
DeckLink Duo	yes	no	yes	no	-	-	-
DeckLink Quad	yes	no	yes	no	-	-	-
DeckLink SDI 4K	yes	no	yes	yes	yes	no	-
DeckLink Studio 4K	yes	yes*	yes	yes	yes	no	-
DeckLink 4K Extreme	yes	yes	yes	yes	yes	no	-
DeckLink 4K Extreme 12G	yes	yes	yes	yes	yes	yes	yes
DeckLink 4K Pro	yes	yes	yes	yes	yes	yes	yes
DeckLink Duo 2	yes	yes	yes	yes	yes	-	-
DeckLink Quad 2	yes	yes	yes	yes	yes	-	-
DeckLink 8K Pro	yes	yes	yes	yes	yes	yes	yes
UltraStudio 4K	yes	yes	yes	yes	yes	no	-
UltraStudio 4K Extreme	yes	yes	yes	yes	yes	yes	yes**
UltraStudio 4K Extreme 3	yes	yes	yes	yes	yes	yes	yes
UltraStudio HD Mini	yes	yes	yes	yes	yes	-	-

- = Video mode not supported for playback

* = SD Only

** = Over PCIe only

Syntax

```
HRESULT      Enable (boolean isExternal);
```

Parameters

Name	Direction	Description
`isExternal`	in	Specifies internal or external keying.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.19.2 IDeckLinkKeyer::SetLevel method

The **SetLevel** method sets the level that the image is blended onto the frame. 0 is no blend, 255 is completely blended onto the frame.

Syntax

```
HRESULT      SetLevel (uint8_t level);
```

Parameters

Name	Direction	Description
`level`	in	The level that the image is to be blended onto the frame.

Return Values

Value	Description
S_OK	Success

2.5.19.3 IDeckLinkKeyer::RampUp method

The **RampUp** method progressively blends in an image over a given number of frames from 0 to 255.

Syntax

```
HRESULT      RampUp (uint32_t numberOfFrames);
```

Parameters

Name	Direction	Description
`numberOfFrames`	in	The number of frames that the image is progressively blended in.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.19.4 IDeckLinkKeyer::RampDown method

The **RampDown** method progressively blends out an image over a given number of frames from 255 to 0.

Syntax

```
HRESULT      RampDown (uint32_t numberOfFrames);
```

Parameters

Name	Direction	Description
`numberOfFrames`	in	The number of frames that the image is progressively blended out.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.19.5 IDeckLinkKeyer::Disable method

The **Disable** method turns off the keyer functionality.

Syntax

```
HRESULT      Disable();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.20 IDeckLinkVideoFrameAncillary Interface

The **IDeckLinkVideoFrameAncillary** object interface represents the ancillary data associated with a video frame. CEA-708 closed-captions are encoded with data bits in the 2 least-signficant-bits of each 10 bit pixel component. These bits are not preserved when capturing in an 8 bit pixel format. To capture or output CEA-708 captions, a 10 bit pixel format such as **bmdFormat10BitYUV** must be used.

Note: The IDeckLinkVideoFrameAncillary object interface is for existing designs or where the ancillary data does not conform to SMPTE 291M type 2 ANC packet format. For new designs with VANC packets, the use of IDeckLinkVideoFrameAncillaryPackets object interface is preferred.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkVideoFrameAncillary** object can be obtained with **IDeckLinkOutput::CreateAncillaryData.**
`IDeckLinkVideoFrame`	`IID_IDeckLinkVideoFrame`	An **IDeckLinkVideoFrameAncillary** object can be obtained from **IDeckLinkVideoFrame::GetAncillaryData.**
`IDeckLinkMutableVideoFrame`	`IID_IDeckLinkMutableVideoFrame`	An **IDeckLinkVideoFrameAncillary** object be set into a video frame using **IDeckLinkMutableVideoFrame::SetAncillaryData.**

Public Member Functions	
Method	**Description**
GetPixelFormat	Gets pixel format of a video frame.
GetDisplayMode	Gets corresponding **BMDDisplayMode** for the selected display mode.
GetBufferForVerticalBlankingLine	Access vertical blanking line buffer.

2.5.20.1 IDeckLinkVideoFrameAncillary::GetPixelFormat method

The **GetPixelFormat** method gets the pixel format of a video frame.

Syntax

```
BMDPixelFormat      GetPixelFormat ();
```

Return Values

Value	Description
PixelFormat	Pixel format of video frame (**BMDPixelFormat**)

2.5.20.2 IDeckLinkVideoFrameAncillary::GetDisplayMode method

The **GetDisplayMode** method returns the corresponding **BMDDisplayMode** for the selected display mode.

Syntax

```
BMDDisplayMode      GetDisplayMode ();
```

Return Values

Value	Description
mode	**BMDDisplayMode** corresponding to the display mode.

2.5.20.3 IDeckLinkVideoFrameAncillary:: GetBufferForVerticalBlankingLine method

The **GetBufferForVerticalBlankingLine** method allows access to a specified vertical blanking line within the ancillary for the associated frame.

Ancillary lines are numbered from one. For NTSC video, the top ancillary lines are numbered starting from four, with lines 1 to 3 referring to the ancillary lines at the bottom of the picture, as per convention.

The pointer returned by **GetBufferForVerticalBlankingLine** is in the same format as the associated active picture data and is valid while the **IDeckLinkVideoFrameAncillary** object interface is valid.

Syntax

```
HRESULT      GetBufferForVerticalBlankingLine
             (uint32_t lineNumber, void* *buffer)
```

Parameters

Name	Direction	Description
`lineNumber`	in	Ancillary line number to access.
`buffer`	out	Pointer into ancillary buffer for requested line or NULL if line number was invalid.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	An invalid ancillary line number was requested

2.5.21 IDeckLinkVideoFrameAncillaryPackets Interface

The **IDeckLinkVideoFrameAncillaryPackets** object interface represents the collection of ancillary data packets associated with a video frame. It is the preferred interface for the capture and output of SMPTE 291M Type 2 VANC packets, replacing legacy **IDeckLinkVideoFrameAncillary** interface.

An **IDeckLinkVideoFrameAncillaryPackets** interface may be obtained from an IDeckLinkVideoFrame object interface using QueryInterface.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkVideoFrame`	`IID_IDeckLinkVideoFrame`	An **IDeckLinkVideoFrameAncillaryPacket** object interface may be obtained from **IDeckLinkVideoFrame** using **QueryInterface**
`IDeckLinkAncillary PacketIterator`	`IID_ IDeckLinkAncillaryPacketIterator`	**IDeckLinkVideoFrameAncillaryPackets ::GetPacketIterator** returns an **IDeckLinkAncillaryPacketIterator** object interface
`IDeckLinkAncillaryPacket`	`IID_IDeckLinkAncillaryPacket`	**IDeckLinkVideoFrameAncillaryPackets ::GetFirstPacketByID** returns an **IDeckLinkAncillaryPacket** object interface

Public Member Functions	
Method	**Description**
GetPacketIterator	Get a iterator that enumerates the available ancillary packets
GetFirstPacketByID	Get the first ancillary packet matching a given DID/SDID pair
AttachPacket	Add an ancillary packet to the video frame
DetachPacket	Remove an ancillary packet from the video frame
DetachAllPackets	Remove all ancillary packets from the video frame.

2.5.21.1 IDeckLinkVideoFrameAncillaryPackets:: GetPacketIterator method

The **GetPacketIterator** method returns an iterator that enumerates the available ancillary packets for a video frame.

Syntax

```
HRESULT     GetPacketIterator
            (IDeckLinkAncillaryPacketIterator **iterator);
```

Parameters

Name	Direction	Description
iterator	out	Pointer to ancillary packet iterator. This object must be released by the caller when no longer required.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	Parameter iterator variable is NULL
E_OUTOFMEMORY	Unable to create iterator

2.5.21.2 IDeckLinkVideoFrameAncillaryPackets:: GetFirstPacketByID method

The **GetFirstPacketByID** method returns the first ancillary packet in the video frame matching a given DID/SDID pair.

Syntax

```
HRESULT     GetFirstPacketByID
            (uint8_t DID, uint8_t SDID,
            IDeckLinkAncillaryPacket **packet);
```

Parameters

Name	Direction	Description
DID	in	Data ID (DID)
SDID	in	Secondary Data ID (SDID)
packet	out	Pointer to ancillary packet. This object must be released by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	Parameter packet variable is NULL

2.5.21.3 IDeckLinkVideoFrameAncillaryPackets:: AttachPacket method

The **AttachPacket** method adds an ancillary packet to the video frame.

Syntax

```
HRESULT      AttachPacket (IDeckLinkAncillaryPacket *packet);
```

Parameters

Name	Direction	Description
packet	in	Ancillary packet to attach

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	Parameter packet variable is NULL or has invalid data stream index
E_OUTOFMEMORY	Unable to allocate memory for packet

2.5.21.4 IDeckLinkVideoFrameAncillaryPackets:: DetachPacket method

The **DetachPacket** method removes an ancillary packet from the video frame.

Syntax

```
HRESULT      DetachPacket (IDeckLinkAncillaryPacket *packet)
```

Parameters

Name	Direction	Description
packet	in	Ancillary packet to detach

Return Values

Value	Description
S_FALSE	Packet not found
S_OK	Success

2.5.21.5 IDeckLinkVideoFrameAncillaryPackets:: DetachAllPackets method

The **DetachAllPackets** method removes all ancillary packets from the video frame.

Syntax

```
HRESULT      DetachAllPackets ();
```

Return Values

Value	Description
S_OK	Success

2.5.22 IDeckLinkAncillaryPacketIterator Interface

The **IDeckLinkAncillaryPacketIterator** object interface is used to enumerate the available ancillary packets in a video frame.

A reference to an IDeckLinkAncillaryIterator object interface for an input video frame may be obtained by calling **GetPacketIterator** on a **IDeckLinkVideoFrameAncillaryPackets** object interface.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkVideoFrame AncillaryPackets`	`IID_IDeckLinkVideoFrame AncillaryPackets`	**IDeckLinkVideoFrameAncillaryPackets ::GetPacketIterator** returns an **IDeckLinkAncillaryIterator** object interface
`IDeckLinkAncillaryPacket`	`IID_ IDeckLinkAncillaryPacket`	**IDeckLinkAncillaryPacketIterator::Next** returns **IDeckLinkAncillaryPacket** interfaces representing each ancillary packet in a video frame

Public Member Functions	
Method	**Description**
Next	Returns an IDeckLinkAncillaryPacket object interface corresponding to an individual ancillary packet.

2.5.22.1 IDeckLinkAncillaryPacketIterator::Next method

The **Next** method creates an object representing an ancillary data packet and assigns the address of the IDeckLinkAncillaryPacket interface of the newly created object to the packet parameter.

Syntax

```
HRESULT      Next (IDeckLinkAncillaryPacket **packet);
```

Parameters

Name	Direction	Description
packet	out	Pointer to **IDeckLinkAncillaryPacket** interface object or NULL when no more ancillary packets are available. This object must be released by the caller when no longer required.

Return Values

Value	Description
S_FALSE	No (more) packets found
S_OK	Success
E_INVALIDARG	Parameter packet variable is NULL

2.5.23 IDeckLinkAncillaryPacket Interface

The **IDeckLinkAncillaryPacket** object interface represents an ancillary data packet within a Video Frame. A reference to an IDeckLinkAncillaryPacket object interface can either be obtained with a known DID/SDID by calling GetFirstPacketByID on a IDeckLinkVideoFrameAncillaryPackets or via the IDeckLinkAncillaryPacketIterator interface.

Developers may subclass **IDeckLinkAncillaryPacket** to implement a specific VANC data packet type.

Related Interfaces

Interface	Interface ID	Description
IDeckLinkAncillary PacketIterator	IID_IDeckLinkAncillary PacketIterator	**IDeckLinkAncillaryPacketIterator::Next** returns **IDeckLinkAncillaryPacket** interfaces representing each ancillary packet in a video frame
IDeckLinkVideoFrame AncillaryPackets	IID_IDeckLinkVideoFrame AncillaryPackets	**IDeckLinkVideoFrameAncillaryPackets ::GetFirstPacketByID** returns an **IDeckLinkAncillaryPacket** object interface

Public Member Functions	
Method	**Description**
GetBytes	Get pointer to ancillary packet data
GetDID	Get Data ID (DID) for ancillary packet
GetSDID	Get Secondary Data ID (SDID) for ancillary packet
GetLineNumber	Get the video frame line number of ancillary packet
GetDataStreamIndex	Get the data stream index for ancillary packet

2.5.23.1 IDeckLinkAncillaryPacket::GetBytes method

The **GetBytes** method allows direct access to the data buffer of the ancillary packet. When subclassing **IDeckLinkAncillaryPacket**, implement **GetBytes** with support of at least one type of **BMDAncillaryPacketFormat**. Specify NULL for either output parameter if unwanted.

Syntax

```
HRESULT      GetBytes (BMDAncillaryPacketFormat format,
             const void **data, uint32_t *size);
```

Parameters

Name	Direction	Description
`format`	in	Requested format of data buffer output (**BMDAncillaryPacketFormat**)
`data`	out	Pointer to ancillary packet data buffer. The pointer is valid while **IDeckLinkAncillaryPacket** object remains valid.
`size`	out	Size of data buffer, in requested format

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_NOTIMPL	Format not implemented

2.5.23.2 IDeckLinkAncillaryPacket::GetDID method

The **GetDID** method returns the Data ID (DID) of the ancillary packet.

Syntax

```
uint8_t      GetDID ();
```

Return Values

Value	Description
DID	Data ID (DID) of the ancillary packet

2.5.23.3 IDeckLinkAncillaryPacket::GetSDID method

The **GetSDID** method returns the SecondaryData ID (SDID) of the ancillary packet.

Syntax

```
uint8_t       GetSDID ();
```

Return Values

Value	Description
SDID	Secondary Data ID (SDID) of the ancillary packet

2.5.23.4 IDeckLinkAncillaryPacket::GetLineNumber method

The **GetLineNumber** method returns the video frame line number of an ancillary packet. When subclassing **IDeckLinkAncillaryPacket** for VANC output, if **GetLineNumber** returns 0, the ancillary packet will be assigned a line automatically determined by the driver.

Syntax

```
uint32_t      GetLineNumber ();
```

Return Values

Value	Description
LineNumber	Video frame line number of the ancillary packet

2.5.23.5 IDeckLinkAncillaryPacket::GetDataStreamIndex method

The **GetDataStreamIndex** method returns a data stream index of the ancillary packet. This function should only return 0 for SD modes. In HD and above, this function will normally return 0 to output the ancillary packet in luma color channel. However this function can return 1 to encode a second data stream in the chroma color channel, but this should only occur when the first data stream is completely full.

Syntax

```
uint8_t       GetDataStreamIndex ();
```

Return Values

Value	Description
DataStreamIndex	Data stream index for the ancillary packet

2.5.24 IDeckLinkTimecode Interface

The **IDeckLinkTimecode** object interface represents a video timecode and provides methods to access the timecode or its components.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkVideoFrameAncillary`	`IID_IDeckLinkVideoFrameAncillary`	**IDeckLinkVideoFrameAncillary::GetTimecode** returns an **IDeckLinkTimecode** object interface

Public Member Functions	
Method	**Description**
GetBCD	Get timecode in BCD
GetComponents	Get timecode components
GetString	Get timecode as formatted string
GetFlags	Get timecode flags
GetTimecodeUserBits	Get timecode user bits.

2.5.24.1 IDeckLinkTimecode::GetBCD method

The **GetBCD** method returns the timecode in Binary Coded Decimal representation.

Syntax

```
BMDTimecodeBCD      GetBCD();
```

Return Values

Value	Description
Timecode	Timecode value in BCD format (See **BMDTimecodeBCD** for details)

2.5.24.2 IDeckLinkTimecode::GetComponents method

The **GetComponents** method returns individual components of the timecode. Specify NULL for any unwanted parameters.

Syntax

```
HRESULT      GetComponents (uint8_t *hours, uint8_t *minutes,
                            uint8_t *seconds, uint8_t *frames);
```

Parameters

Name	Direction	Description
`hours`	out	Hours component of timecode
`minutes`	out	Minutes component of timecode
`seconds`	out	Seconds component of timecode
`frames`	out	Frames component of timecode

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.24.3 IDeckLinkTimecode::GetString method

The **GetString** method returns the timecode formatted as a standard timecode string.

Syntax

```
HRESULT      GetString(string *timecode);
```

Parameters

Name	Direction	Description
`timecode`	out	Timecode formatted as a standard timecode string: “HH:MM:SS:FF”. This allocated string must be freed by the caller when no longer required

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.24.4 IDeckLinkTimecode::GetFlags method

The **GetFlags** method returns the flags accompanying a timecode.

Syntax

```
HRESULT      BMDTimecodeFlags    GetFlags();
```

Return Values

Value	Description
timecodeFlags	Timecode flags (see **BMDTimecodeFlags** for details)

2.5.24.5 IDeckLinkTimecode::GetTimecodeUserBits method

The **GetTimecodeUserBits** method returns the timecode user bits.

Syntax

```
HRESULT      GetTimecodeUserBits (BMDTimecodeUserBits *userBits);
```

Parameters

Name	Direction	Description
`userBits`	out	The user bits.

Return Values

Value	Description
E_POINTER	The userBits parameter is NULL.
S_OK	Success

2.5.25 IDeckLinkScreenPreviewCallback Interface

The **IDeckLinkScreenPreviewCallback** object interface is a callback class which is called to facilitate updating of an on-screen preview of a video stream being played or captured.

An object with the **IDeckLinkScreenPreviewCallback** object interface may be registered as a callback with the **IDeckLinkInput** or **IDeckLinkOutput** interfaces.

During playback or capture, frames will be delivered to the preview callback. A dedicated preview thread waits for the next available frame before calling the callback. The frame delivery rate may be rate limited by the preview callback - it is not required to maintain full frame rate and missing frames in preview will have no impact on capture or playback.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkInput`	`IID_IDeckLinkInput`	An **IDeckLinkScreenPreviewCallback** object interface may be registered with **IDeckLinkInput::SetScreenPreviewCallback**
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkScreenPreviewCallback** object interface may be registered with **IDeckLinkOutput::SetScreenPreviewCallback**

Public Member Functions	
Method	**Description**
DrawFrame	Called when a new frame is available for the preview display

2.5.25.1 IDeckLinkScreenPreviewCallback::DrawFrame method

The **DrawFrame** method is called on every frame boundary while scheduled playback is running.

For example: Scheduled NTSC which runs at 29.97 frames per second, will result in the preview callback's DrawFrame() method being called 29.97 times per second while scheduled playback is running.

The return value (required by COM) is ignored by the caller.

Note: If the frame to be drawn to the preview hasn't changed since the last time the callback was called, the frame parameter will be NULL.

Syntax

```
HRESULT      DrawFrame(IDeckLinkVideoFrame *theFrame);
```

Parameters

Name	Direction	Description
`theFrame`	in	Video frame to preview

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.26 IDeckLinkGLScreenPreviewHelper Interface

The **IDeckLinkGLScreenPreviewHelper** object interface may be used with a simple **IDeckLinkScreenPreviewCallback** implementation
to provide OpenGL based preview rendering which is decoupled from the incoming or outgoing video stream being previewed.

A reference to an **IDeckLinkGLScreenPreviewHelper** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateOpenGLScreenPreviewHelper** on other platforms.

Typical usage of **IDeckLinkGLScreenPreviewHelper** is as follows:

- Configure an OpenGL context as an orthographic projection using code similar to the following:

```
glViewport(0, 0, (GLsizei)newSize.width, (GLsizei)newSize.height);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glOrtho(-1.0, 1.0, -1.0, 1.0, -1.0, 1.0);
glMatrixMode(GL_MODELVIEW);
```

- Create an **IDeckLinkGLScreenPreviewHelper** object interface using **CoCreateInstance** or **CreateOpenGLScreenPreviewHelper**
  Call **IDeckLinkGLScreenPreviewHelper::InitializeGL** from the OpenGL context
- When repainting the **OpenGL** context, call **IDeckLinkGLScreenPreviewHelper::PaintGL**. The preview image will be drawn between (-1,-1) and (1,1) in the GL space.
- Add any graphical overlays on the preview window as desired.
- Create a subclass of **IDeckLinkScreenPreviewCallback** which calls **IDeckLinkGLScreenPreviewHelper::SetFrame** from **IDeckLinkScreenPreviewCallback::DrawFrame**
- Register an instance of the **IDeckLinkScreenPreviewCallback** subclass with **IDeckLinkInput::SetScreenPreviewCallback** or **IDeckLinkOutput::SetScreenPreviewCallback** as appropriate.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkScreenPreview`	`IID_IDeckLinkScreenPreview`	**IDeckLinkGLScreenPreviewHelper::SetFrame** may be called from **IDeckLinkScreenPreview::DrawFrame**

Public Member Functions	
Method	**Description**
InitializeGL	Initialize GL previewing
PaintGL	Repaint the GL preview
SetFrame	Set the preview frame to display on the next PaintGL call
Set3DPreviewFormat	Set the 3D preview format.

2.5.26.1 IDeckLinkGLScreenPreviewHelper::InitializeGL method

The **InitializeGL** method should be called from the preview OpenGL context during initialization of that context.

Syntax

```
HRESULT      InitializeGL();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.26.2 IDeckLinkGLScreenPreviewHelper::PaintGL method

The **PaintGL** method should be called from the preview OpenGL context whenever the preview frame needs to be repainted. Frames to be displayed should be provided to **IDeckLinkGLScreenPreviewHelper::SetFrame**.

PaintGL and **SetFrame** allow OpenGL updates to be decoupled from new frame availability.

Syntax

```
HRESULT      PaintGL();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.26.3 IDeckLinkGLScreenPreviewHelper::SetFrame method

The **SetFrame** method is used to set the preview frame to display on the next call to **IDeckLinkGLScreenPreviewHelper::PaintGL**.

Depending on the rate and timing of calls to **SetFrame** and **PaintGL**, some frames may not be displayed or may be displayed multiple times.

Syntax

```
HRESULT      SetFrame(IDeckLinkVideoFrame *theFrame)
```

Parameters

Name	Direction	Description
`theFrame`	in	Video frame to preview

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.26.4 IDeckLinkGLScreenPreviewHelper::Set3DPreviewFormat

The **Set3DPreviewFormat** method is used to set the 3D preview format.

Syntax

```
HRESULT      Set3DPreviewFormat(BMD3DPreviewFormat *previewFormat);
```

Parameters

Name	Direction	Description
previewFormat	in	The 3D preview format. See the Linked frame preview format (**BMD3DPreviewFormat**) section for more details.

Return Values

Value	Description
S_OK	Success

2.5.27 IDeckLinkCocoaScreenPreviewCallback Interface

The **IDeckLinkCocoaScreenPreviewCallback** object interface is a cocoa callback class which is called to facilitate updating of an on-screen preview of a video stream being played or captured.

An **IDeckLinkCocoaScreenPreviewCallback** object can be created by calling CreateCocoaScreenPreview. This object can registered as a callback with **IDeckLinkInput::SetScreenPreviewCallback** or **IDeckLinkOutput::SetScreenPreviewCallback** as appropriate.

During playback or capture, frames will be delivered to the preview callback. A dedicated preview thread waits for the next available frame before calling the callback. The frame delivery rate may be rate limited by the preview callback - it is not required to maintain full frame rate and missing frames in preview will have no impact on capture or playback.

Related Interfaces

Interface	Interface ID	Description
IDeckLinkInput	IID_IDeckLinkInput	An **IDeckLinkCocoaScreenPreviewCallback** object interface may be registered with **IDeckLinkInput::SetScreenPreviewCallback**
IDeckLinkOutput	IID_IDeckLinkOutput	An **IDeckLinkCocoaScreenPreviewCallback** object interface may be registered with **IDeckLinkOutput::SetScreenPreviewCallback**

2.5.28 IDeckLinkDX9ScreenPreviewHelper Interface

The **IDeckLinkDX9ScreenPreviewHelper** object interface may be used with a simple **IDeckLinkScreenPreviewCallback** implementation to provide DirectX based preview rendering which is decoupled from the incoming or outgoing video stream being previewed.

A reference to an **IDeckLinkDX9ScreenPreviewHelper** object is obtained from **CoCreateInstance**.

Typical usage of **IDeckLinkDX9ScreenPreviewHelper** is as follows:

- Create an **IDeckLinkDX9ScreenPreviewHelper** object interface using CoCreateInstance.
- If 3D preview is required, call **IDeckLinkDX9ScreenPreviewHelper::Set3DPreviewFormat**
- Setup Direct 3D parameters:

```
D3DPRESENT_PARAMETERS       d3dpp;
IDirect3DDevice9*           dxDevice;
d3dpp.BackBufferFormat = D3DFMT_UNKNOWN;
d3dpp.BackBufferCount = 2;
d3dpp.Windowed = TRUE;
d3dpp.SwapEffect = D3DSWAPEFFECT_DISCARD;
d3dpp.hDeviceWindow = hwnd;
d3dpp.PresentationInterval = D3DPRESENT_INTERVAL_DEFAULT;
```

- Create a new device: CreateDevice(D3DADAPTER_DEFAULT, D3DDEVTYPE_HAL, hwnd, D3DCREATE_HARDWARE_VERTEXPROCESSING | D3DCREATE_MULTITHREADED, &d3dpp, &dxDevice);
- Call **IDeckLinkDX9ScreenPreviewHelper::Initialize** (dxDevice)
- When repainting, call the following methods:

```
dxDevice->BeginScene();
IDeckLinkDX9ScreenPreviewHelper::Render();
dxDevice->EndScene();
```

- Create a subclass of **IDeckLinkScreenPreviewCallback** which calls **IDeckLinkDX9ScreenPreviewHelper::SetFrame** from **IDeckLinkScreenPreviewCallback::DrawFrame**.
- Register an instance of the **IDeckLinkScreenPreviewCallback** subclass with **IDeckLinkInput::SetScreenPreviewCallback** or **IDeckLinkOutput::SetScreenPreviewCallback** as appropriate.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkScreenPreview`	`IID_IDeckLinkScreenPreview`	**IDeckLinkDX9ScreenPreviewHelper::SetFrame** may be called from **IDeckLinkScreenPreview::DrawFrame**

Public Member Functions	
Method	**Description**
Initialize	Initialize DirectX previewing.
Render	Repaint the DirectX preview.
SetFrame	Set the preview frame for display.
Set3DPreviewFormat	Set the 3D preview format.

2.5.28.1 IDeckLinkDX9ScreenPreviewHelper::Initialize method

The **Initialize** method sets the IDirect3DDevice9 object to be used by the DeckLink API's preview helper.

Syntax

```
HRESULT      Initialize (void *device);
```

Parameters

Name	Direction	Description
device	in	The IDirect3DDevice9 object

Return Values

Value	Description
S_OK	Success

2.5.28.2 IDeckLinkDX9ScreenPreviewHelper::Render method

The **Render** method should be called whenever the preview frame needs to be repainted. The frames to be displayed should be provided to **IDeckLinkDX9ScreenPreviewHelper::SetFrame**.

Syntax

```
HRESULT      Render (RECT *rc)
```

Parameters

Name	Direction	Description
rc	in	The display surface rectangle. If rc is NULL, the whole view port / surface is used. If the rc dimensions have changed, the display texture will be resized.

Return Values

Value	Description
S_OK	Success

2.5.28.3 IDeckLinkDX9ScreenPreviewHelper::SetFrame method

The **SetFrame** method will set a 2D or 3D **IDeckLinkVideoFrame** into a texture. This method is used to set the preview frame to display on the next call to **IDeckLinkDX9ScreenPreviewHelper::Render**. Depending on the rate and timing of calls to **SetFrame** and **Render**, some frames may not be displayed or may be displayed multiple times.

Syntax

```
HRESULT      SetFrame (IDeckLinkVideoFrame *primaryFrame);
```

Parameters

Name	Direction	Description
`primaryFrame`	in	The video frame to preview.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.28.4 IDeckLinkDX9ScreenPreviewHelper:: Set3DPreviewFormat method

The **Set3DPreviewFormat** method is used to set the 3D preview format.

Syntax

```
HRESULT      Set3DPreviewFormat (BMD3DPreviewFormat previewFormat);
```

Parameters

Name	Direction	Description
`previewFormat`	in	The 3D preview format. See the 'Frame preview format' section (**BMD3DPreviewFormat**) for more details.

Return Values

Value	Description
S_OK	Success

2.5.29 IDeckLinkDeckControl Interface

The **IDeckLinkDeckControl** object interface provides the capability to control a deck via the RS422 port (if available) of a DeckLink device.

An **IDeckLinkDeckControl** object interface can be obtained from the **IDeckLink** interface using **QueryInterface**.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkDeckControl`	`IID_IDeckLinkDeckControl`	An **IDecklinkDeckControl** object interface may be obtained from **IDeckLink** using **QueryInterface**.
`IDeckLinkDeckControlStatusCallback`	`IID_IDeckLinkDeckControlStatusCallback`	An **IDeckLinkDeckControlStatusCallback** object interface may be registered with **IDeckLinkDeckControl::SetCallback**.

Public Member Functions	
Method	**Description**
Open	Open a connection to the deck.
Close	Close the connection to the deck.
GetCurrentState	Get the current state of the deck.
SetStandby	Put the deck into standby mode.
SendCommand	Send a custom command to the deck.
Play	Send a play command to the deck.
Stop	Send a stop command to the deck.
TogglePlayStop	Toggle between play and stop mode.
Eject	Send an eject command to the deck.
GoToTimecode	Set the deck to go the specified timecode on the tape.
FastForward	Send a fast forward command to the deck.
Rewind	Send a rewind command to the deck.
StepForward	Send a step forward command to the deck.
StepBack	Send a step back command to the deck.
Jog	Send a jog forward / reverse command to the deck.
Shuttle	Send a shuttle forward / reverse command to the deck.
GetTimecodeString	Get a timecode from deck in string format.
GetTimecode	Get a timecode from deck in IDeckLinkTimeCode format.
GetTimecodeBCD	Get a timecode from deck in BMDTimecodeBCD format.
SetPreroll	Set the preroll period.
GetPreroll	Get the preroll period.
SetCaptureOffset	Set the field accurate capture timecode offset.
GetCaptureOffset	Current capture timecode offset
SetExportOffset	Set the field accurate export timecode offset.

Method	Description
GetExportOffset	Get the current setting of the field accurate export timecode offset.
GetManualExportOffset	Get the recommended delay fields of the current deck.
StartExport	Start an export to tape.
StartCapture	Start a capture.
GetDeviceID	Get deck device ID.
Abort	Stop current deck operation.
CrashRecordStart	Send a record command to the deck.
CrashRecordStop	Send a stop record command to the deck.
SetCallback	Set a deck control status callback.

2.5.29.1 IDeckLinkDeckControl::Open method

The **Open** method configures a deck control session and opens a connection to a deck. This command will fail if a RS422 serial port is not available on the DeckLink device.

The application should wait for a **IDeckLinkDeckControlStatusCallback::DeckControlStatusChanged** callback notification with the **bmdDeckControlStatusDeckConnected** bit set before using the rest of the deck control functionality.

Syntax

```
HRESULT      Open (BMDTimeScale timeScale, BMDTimeValue timeValue,
                   boolean timecodeIsDropFrame,
                   BMDDeckControlError *error)
```

Parameters

Name	Direction	Description
`timeScale`	in	The time scale.
`timeValue`	in	The time value in units of BMDTimeScale.
`timecodeIsDropFrame`	in	Timecode is drop frame (TRUE) or a non drop frame (FALSE).
`error`	out	The error code from the deck - see BMDDeckControlError for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.2 IDeckLinkDeckControl::Close method

The **Close** method will optionally place the deck in standby mode before closing the connection.

Syntax

```
HRESULT      Close (boolean standbyOn)
```

Parameters

Name	Direction	Description
`standbyOn`	in	Place the deck into standby mode (TRUE) before disconnection.

Return Values

Value	Description
S_OK	Success

2.5.29.3 IDeckLinkDeckControl::GetCurrentState method

The **GetCurrentState** method will get the current state of the deck.

Syntax

```
HRESULT      GetCurrentState
             (BMDDeckControlMode *mode,
             BMDDeckControlVTRControlState *vtrControlState,
             BMDDeckControlStatusFlags *flags);
```

Parameters

Name	Direction	Description
`mode`	out	The deck control mode - see **BMDDeckControlMode** for details.
`vtrControlState`	out	The deck control state - see **BMDDeckControlVTRControlState** for details.
`flags`	out	The deck control status flags - see **BMDDeckControlStatusFlags** for details.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.4 IDeckLinkDeckControl::SetStandby method

The **SetStandby** method will send a “set standby” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      SetStandby (boolean standbyOn);
```

Parameters

Name	Direction	Description
`standbyOn`	in	Set standby on (TRUE) , or set standby off (FALSE)

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.29.5 IDeckLinkDeckControl::SendCommand method

The **SendCommand** method will send a custom command to the deck. A custom command operation cannot occur if there is an export-to-tape, capture or a custom command operation in progress. The supplied custom command must conform to the Sony 9 Pin protocol and must not include the checksum byte. It will be generated by this interface and added to the command. The deck’s response (minus the checksum) is stored in the provided buffer.

Syntax

```
HRESULT      SendCommand
             (uint8_t *inBuffer, uint32_t inBufferSize,
             uint8_t *outBuffer, uint32_t *outDataSize,
             uint32_t outBufferSize, BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`inBuffer`	in	The buffer containing the command packet to transmit.
`inBufferSize`	in	The size of the buffer containing the command packet to transmit.
`outBuffer`	out	The buffer to contain the response packet.
`outDataSize`	out	The size of the response data.
`outBufferSize`	out	The size of the buffer that will contain the response packet.
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_INVALIDARG	One or more parameters are invalid.
E_UNEXPECTED	A previous custom command is still being processed.
E_FAIL	Failure - check error parameter
S_OK	Success

2.5.29.6 IDeckLinkDeckControl::Play method

The **Play** method will send a “play” command to the deck. The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      Play (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
error	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.7 IDeckLinkDeckControl::Stop method

The **Stop** method will send a “stop” command to the deck. The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      Stop (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
error	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.8 IDeckLinkDeckControl::TogglePlayStop method

The **TogglePlayStop** method will send a “play” command to the deck, if the deck is currently paused or stopped. If the deck is currently playing, a “pause” command will be sent to the deck. The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      TogglePlayStop (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
error	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.9 IDeckLinkDeckControl::Eject method

The **Eject** method will send an “eject tape” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      Eject (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
error	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.10 IDeckLinkDeckControl::GoToTimecode method

The **GoToTimecode** method will send a “go to timecode” command to the deck.

Syntax

```
HRESULT      GoToTimecode (BMDTimecodeBCD timecode,
                           BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`timecode`	in	The timecode to go to.
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.11 IDeckLinkDeckControl::FastForward method

The **FastForward** method will send a “fast forward” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      FastForward (boolean viewTape,
                          BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`viewTape`	in	View the tape (TRUE) or enable automatic selection of “tape view” or “end to end view” (FALSE)
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.12 IDeckLinkDeckControl::Rewind method

The **Rewind** method will send a “rewind” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      Rewind (boolean viewTape, BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`viewTape`	in	View the tape (TRUE) or enable automatic selection of “tape view” or “end to end view” (FALSE)
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.13 IDeckLinkDeckControl::StepForward method

The **StepForward** method will send a “step forward” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      StepForward (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.14 IDeckLinkDeckControl::StepBack method

The **StepBack** method will send a “step back” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      StepBack (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.15 IDeckLinkDeckControl::Jog method

The **Jog** method will send a “jog playback” command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      Jog (double rate, BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`rate`	in	The rate at which to jog playback. A value greater than 0 will enable forward playback, value less than 0 will enable reverse playback. The rate range is from -50.0 to 50.0
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.16 IDeckLinkDeckControl::Shuttle method

The **Shuttle** method will send a “shuttle” playback command to the deck.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      Shuttle (double rate, BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`rate`	in	The rate at which to shuttle playback. A value greater than 0 will enable forward playback, a value less than 0 will enable reverse playback. The rate range is from -50.0 to 50.0
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.17 IDeckLinkDeckControl::GetTimecodeString method

The **GetTimecodeString** method will return the current timecode in string format.

Syntax

```
HRESULT      GetTimecodeString
             (string currentTimeCode, BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`currentTimeCode`	out	The current timecode in string format.
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.18 IDeckLinkDeckControl::GetTimecode method

The **GetTimecode** method will return the current timecode in **IDeckLinkTimecode** format.

Syntax

```
HRESULT      GetTimecode (IDeckLinkTimecode currentTimecode,
                          BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
currentTimeCode	out	The current timecode in **IDeckLinkTimecode** format.
error	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.19 IDeckLinkDeckControl::GetTimecodeBCD method

The **GetTimecodeBCD** method will return the current timecode in BCD format.

Syntax

```
HRESULT      GetTimecodeBCD (BMDTimecodeBCD *currentTimecode,
                             BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
currentTimeCode	out	The timecode in BCD format.
error	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.20 IDeckLinkDeckControl::SetPreroll method

The **SetPreroll** method will set the preroll time period.

Syntax

```
HRESULT      SetPreroll (uint32_t prerollSeconds);
```

Parameters

Name	Direction	Description
prerollSeconds	in	The preroll period in seconds to set.

Return Values

Value	Description
S_OK	Success

2.5.29.21 IDeckLinkDeckControl::GetPreroll method

The **GetPreroll** method will get the preroll period setting.

Syntax

```
HRESULT      GetPreroll (uint32_t *prerollSeconds);
```

Parameters

Name	Direction	Description
prerollSeconds	out	The current preroll period.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.22 IDeckLinkDeckControl::SetCaptureOffset method

The capture offset may be used to compensate for a deck specific offset between the inpoint and the time at which the capture starts.

Syntax

```
HRESULT      SetCaptureOffset (int32_t captureOffsetFields);
```

Parameters

Name	Direction	Description
captureOffsetFields	in	The timecode offset to set in fields.

Return Values

Value	Description
S_OK	Success

2.5.29.23 IDeckLinkDeckControl::GetCaptureOffset method

The **GetCaptureOffset** method will return the current setting of the field accurate capture timecode offset in fields.

Syntax

```
HRESULT      GetCaptureOffset (int32_t *captureOffsetFields);
```

Parameters

Name	Direction	Description
captureOffsetFields	out	The current timecode offset in fields.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.24 IDeckLinkDeckControl::SetExportOffset method

The **SetExportOffset** method will set the current export timecode offset in fields. This method permits fine control of the timecode offset to tailor for the response of an individual deck by adjusting the number of fields prior to the in or out point where an export will begin or end.

Syntax

```
HRESULT      SetExportOffset (int32_t exportOffsetFields);
```

Parameters

Name	Direction	Description
exportOffsetFields	in	The timecode offset in fields.

Return Values

Value	Description
S_OK	Success

2.5.29.25 IDeckLinkDeckControl::GetExportOffset method

The **GetExportOffset** method will return the current setting of the export offset in fields.

Syntax

```
HRESULT      GetExportOffset (int32_t * exportOffsetFields);
```

Parameters

Name	Direction	Description
exportOffsetFields	out	The current timecode offset in fields.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.26 IDeckLinkDeckControl::GetManualExportOffset method

The **GetManualExportOffset** method will return the manual export offset for the current deck. This is only applicable for manual exports and may be adjusted with the main export offset if required.

Syntax

```
HRESULT      GetManualExportOffset
             (int32_t * deckManualExportOffsetFields);
```

Parameters

Name	Direction	Description
deckManualExportOffsetFields	out	The current timecode offset.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.27 IDeckLinkDeckControl::StartExport method

The **StartExport** method starts an export to tape operation using the given parameters. Prior to calling this method, the output interface should be set up as normal (refer to the **Playback** and **IDeckLinkOutput** interface sections). **StartScheduledPlayback** should be called in the **bmdDeckControlPrepareForExportEvent** event in **IDeckLinkDeckControlStatusCallback::DeckControlEventReceived** callback. The callback object should be set using **IDeckLinkDeckControl::SetCallback**. A connection to the deck should then be opened using **IDeckLinkDeckControl::Open**. The preroll period can be set using **IDeckLinkDeckControl::SetPreroll** and an offset period set using **IDeckLinkDeckControl::SetExportOffset**.

After **StartExport** is called, the export will commence when the current time code equals the “inTimecode”. Scheduled frames are exported until the current timecode equals the “outTimecode”. During this period the **IDeckLinkDeckControlStatusCallback** will be called when deck control events occur.

At the completion of the export operation the **bmdDeckControlExportCompleteEvent** in the **IDeckLinkDeckControlStatusCallback::DeckControlEventReceived** will occur several frames from the “outTimecode”.
Resources may be released at this point or another export may be commenced.

Syntax

```
HRESULT      StartExport
             (BMDTimecodeBCD inTimecode, BMDTimecodeBCD outTimecode,
             BMDDeckControlExportModeOpsFlags exportModeOps,
             BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`inTimecode`	in	The timecode to start the export sequence.
`outTimecode`	in	The timecode to stop the export sequence.
`exportModeOps`	in	The export mode operations - see **BMDDeckControlExportModeOpsFlags** for details.
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.28 IDeckLinkDeckControl::StartCapture method

The **StartCapture** method starts a capture operation using the given parameters. Prior to calling this method, the input interface should be set up as normal (refer to the **Capture** and **IDeckLinkInput** interface sections), **IDeckLinkDeckControl** should be configured (see description below) and a connection to the deck established using **IDeckLinkDeckControl::Open**.

A callback object should be set using **IDeckLinkDeckControl::SetCallback** and an offset period set using **IDeckLinkDeckControl::SetCaptureOffset**.

After **StartCapture** is called, the application must wait until the **bmdDeckControlPrepareForCaptureEvent** event is received via **IDeckLinkDeckControlStatusCallback::DeckControlEventReceived** callback. Reception of that event signals that the serial timecodes attached to the **IDeckLinkVideoFrame** objects (received via **IDeckLinkInputCallback::VideoInputFrameArrived**) can be used to determine if the frame is between the inTimecode and outTimecode timecodes.

The application must take into account that the serial timecode values should be adjusted by the value set using **IDeckLinkDeckControl::SetCaptureOffset**.

During this period **IDeckLinkDeckControlStatusCallback** will be called when deck control events occur.

At the completion of the capture operation the **bmdDeckControlCaptureCompleteEvent** event in the **IDeckLinkDeckControlStatus Callback::DeckControlEventReceived** method will occur several frames from the “outTimecode”. Resources may be released at this point. **IDeckLinkDeckControl** will return to VTR control mode.

Syntax

```
HRESULT     StartCapture
            (boolean useVITC, BMDTimecodeBCD inTimecode,
            BMDTimecodeBCD outTimecode,
            BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`useVITC`	in	If true use VITC as the source of timecodes.
`inTimecode`	in	The timecode to start the capture sequence.
`outTimecode`	in	The timecode to stop the capture sequence.
`error`	out	Error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.29 IDeckLinkDeckControl::GetDeviceID method

The **GetDeviceID** method gets the device ID returned by the deck.
The **IDeckLinkDeckControl** must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      GetDeviceID (uint16_t *deviceId,
                          BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`deviceId`	out	The code for the device model.
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.5.29.30 IDeckLinkDeckControl::Abort method

The **Abort** operation is synchronous. Completion is signaled with a **bmdDeckControlAbortedEvent** event.

Syntax

```
HRESULT      Abort (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.29.31 IDeckLinkDeckControl::CrashRecordStart method

The **CrashRecordStart** method sets the deck to record.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      CrashRecordStart (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.32 IDeckLinkDeckControl::CrashRecordStop method

The **CrashRecordStop** method stops the deck record operation.
The **IDeckLinkDeckControl** object must be in VTR control mode for this command to succeed.

Syntax

```
HRESULT      CrashRecordStop (BMDDeckControlError *error);
```

Parameters

Name	Direction	Description
`error`	out	The error code sent by the deck - see **BMDDeckControlError** for details.

Return Values

Value	Description
E_FAIL	Failure - check error parameter.
S_OK	Success
E_INVALIDARG	The parameter is invalid.

2.5.29.33 IDeckLinkDeckControl::SetCallback method

The **SetCallback** method installs a callback object to be called when deck control events occur.

Syntax

```
HRESULT      SetCallback
             (IDeckLinkDeckControlStatusCallback *callback);
```

Parameters

Name	Direction	Description
`callback`	in	The callback object implementing the **IDeckLinkDeckControlStatusCallback** object interface

Return Values

Value	Description
S_OK	Success

2.5.30 IDeckLinkDeckControlStatusCallback Interface

The **IDeckLinkDeckControlStatusCallback** object interface is a callback class which is called when the Deck control status has changed.

An object with the **IDeckLinkDeckControlStatusCallback** object interface may be registered as a callback with the **IDeckLinkDeckControl** interface.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkDeckControl`	`IID_IDeckLinkDeckControl`	An **IDeckLinkDeckControlStatusCallBack** object interface may be registered with **IDeckLinkDeckControl::SetCallback**

Public Member Functions	
Method	**Description**
TimecodeUpdate	Called when there is a change to the timecode.
VTRControlStateChanged	Called when the control state of the deck changes.
DeckControlEventReceived	Called when a deck control event occurs.
DeckControlStatusChanged	Called when deck control status has changed.

2.5.30.1 IDeckLinkDeckControlStatusCallback:: TimecodeUpdate method

The **TimecodeUpdate** method is called when there is a change to the timecode.

Timecodes may be missed when playing at non 1x speed. This method will not be called during capture, and the serial timecode attached to each frame delivered by the API should be used instead.

Syntax

```
HRESULT      TimecodeUpdate (BMDTimecodeBCD currentTimecode);
```

Parameters

Name	Direction	Description
`currentTimecode`	in	The current timecode.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.30.2 IDeckLinkDeckControlStatusCallback:: VTRControlStateChanged method

The **VTRControlStateChanged** method is called when there is a change in the deck control state. Refer to **BMDDeckControlVTRControlState** for the possible states. This method is only called while in VTR control mode.

Syntax

```
HRESULT      VTRControlStateChanged
             (BMDDeckControlVTRControlState newState,
             BMDDeckControlError error);
```

Parameters

Name	Direction	Description
`newState`	in	The new deck control state - see **BMDDeckControlVTRControlState** for details.
`error`	in	The deck control error code.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.30.3 IDeckLinkDeckControlStatusCallback:: DeckControlEventReceived method

The **DeckControlEventReceived** method is called when a deck control event occurs.

Syntax

```
HRESULT      DeckControlEventReceived
             (BMDDeckControlEvent event, BMDDeckControlError error);
```

Parameters

Name	Direction	Description
`event`	in	The deck control event that has occurred - see **BMDDeckControlEvent** for details.
`error`	in	The deck control error that has occurred.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.30.4 IDeckLinkDeckControlStatusCallback:: DeckControlStatusChanged method

The **DeckControlStatusChanged** method is called when the deck control status has changed.

Syntax

```
HRESULT      DeckControlStatusChanged
             (BMDDeckControlStatusFlags flags, uint32_t mask);
```

Parameters

Name	Direction	Description
`flags`	in	The deck control current status - see **BMDDeckControlStatusFlags** for details.
`mask`	in	The deck control status event flag(s) that has changed.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.31 IDeckLinkDiscovery Interface

The **IDeckLinkDiscovery** object interface is used to install or remove the callback for receiving DeckLink device discovery notifications.
A reference to an **IDeckLinkDiscovery** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateDeckLinkDiscoveryInstance** on other platforms.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkDevice NotificationCallback`	`IID_IDeckLinkDevice NotificationCallback`	A device notification callback can be installed with **IDeckLinkDiscovery::InstallDeviceNotifications** or uninstalled with **IDeckLinkDiscovery::UninstallDeviceNotifications**

Public Member Functions	
Method	**Description**
InstallDeviceNotifications	Install DeckLink device notifications callback
UninstallDeviceNotifications	Remove DeckLink device notifications callback

2.5.31.1 IDeckLinkDiscovery::InstallDeviceNotifications method

The **InstallDeviceNotifications** method installs the **IDeckLinkDeviceNotificationCallback** callback which will be called when a new DeckLink device becomes available.

Syntax

```
HRESULT      InstallDeviceNotifications
             (IDeckLinkDeviceNotificationCallback* deviceCallback);
```

Parameters

Name	Direction	Description
`deviceCallback`	in	Callback object implementing the **IDeckLinkDeviceNotificationCallback** object interface.

Return Values

Value	Description
E_INVALIDARG	The parameter variable is NULL
E_FAIL	Failure
S_OK	Success

2.5.31.2 IDeckLinkDiscovery:: UninstallDeviceNotifications method

The **UninstallDeviceNotifications** method removes the DeckLink device notifications callback. When this method returns, it guarantees there are no ongoing callbacks to the **IDeckLinkDeviceNotificationCallback** instance.

Syntax

```
HRESULT      UninstallDeviceNotifications (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.32 IDeckLinkDeviceNotificationCallback

The **IDeckLinkDeviceNotificationCallback** object interface is callback which is called when a DeckLink device arrives or is removed.

Public Member Functions	
Method	**Description**
DeckLinkDeviceArrived	A DeckLink device has arrived.
DeckLinkDeviceRemoved	A DeckLink device has been removed.

2.5.32.1 IDeckLinkDeviceNotificationCallback:: DeckLinkDeviceArrived method

The **DeckLinkDeviceArrived** method is called when a new DeckLink device becomes available. This method will be called on an API private thread.

This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      DeckLinkDeviceArrived (IDeckLink* deckLinkDevice);
```

Parameters

Name	Direction	Description
deckLinkDevice	in	DeckLink device. The **IDeckLink** reference will be released when the callback returns. To hold on to it beyond the callback, call **AddRef**. Your application then owns the **IDeckLink** reference and is responsible for managing the **IDeckLink** object's lifetime. The reference can be released at any time (including in the **DeckLinkDeviceRemoved** callback) by calling **Release**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.32.2 IDeckLinkDeviceNotificationCallback:: DeckLinkDeviceRemoved method

The **DeckLinkDeviceRemoved** method is called when a DeckLink device is disconnected. This method will be called on an API private thread.

This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      DeckLinkDeviceRemoved (IDeckLink* deckLinkDevice);
```

Parameters

Name	Direction	Description
`deckLinkDevice`	in	DeckLink device.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.33 IDeckLinkNotification Interface

The **IDeckLinkNotification** object interface is used to install or remove the callback for receiving DeckLink device notifications.

An **IDeckLinkNotification** object interface may be obtained from **IDeckLink** using **QueryInterface**.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	An **IDeckLinkNotification** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkNotificationCallback`	`IID_IDeckLinkNotification Callback`	An **IDeckLinkNotificationCallback** object can be subscribed using **IDeckLinkNotification::Subscribe** or unsubscribed using **IDeckLinkNotification::Unsubscribe**

Public Member Functions	
Method	**Description**
Subscribe	Subscribe a notification. Please see **BMDNotifications** for more details.
Unsubscribe	Unsubscribe a notification

2.5.33.1 IDeckLinkNotification::Subscribe method

The **Subscribe** method registers a callback object for a given topic.

Syntax

```
HRESULT      Subscribe (BMDNotifications topic,
                        IDeckLinkNotificationCallback *theCallback);
```

Parameters

Name	Direction	Description
`topic`	in	The notification event type.
`theCallback`	in	The callback object implementing the **IDeckLinkNotificationCallback** object interface.

Return Values

Value	Description
E_INVALIDARG	The callback parameter variable is NULL
E_FAIL	Failure
S_OK	Success

2.5.33.2 IDeckLinkNotification::Unsubscribe method

The **Unsubscribe** method removes a notification event type from a callback object.

Syntax

```
HRESULT      Unsubscribe (BMDNotifications topic,
                          IDeckLinkNotificationCallback *theCallback);
```

Parameters

Name	Direction	Description
`topic`	in	The notification event type.
`theCallback`	in	The callback object implementing the **IDeckLinkNotificationCallback** object interface.

Return Values

Value	Description
E_INVALIDARG	The callback parameter variable is NULL
E_FAIL	Failure
S_OK	Success

2.5.34 IDeckLinkNotificationCallback Interface

The **IDeckLinkNotificationCallback** object interface is used to notify the application about a subscribed event.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkNotification`	`IID_IDeckLinkNotification`	An **IDeckLinkNotificationCallback** object can be subscribed using **IDeckLinkNotification::Subscribe** An **IDeckLinkNotificationCallback** object can be unsubscribed using **IDeckLinkNotification::Unsubscribe**

Public Member Functions	
Method	**Description**
Notify	Called when a subscribed notification event has occurred.

2.5.34.1 IDeckLinkNotificationCallback::Notify method

The **Notify** method is called when subscribed notification occurs.

This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      Notify (BMDNotifications topic, uint64_t param1,
                     uint64_t param2);
```

Parameters

Name	Direction	Description
`topic`	in	The type of notification. Please see **BMDNotifications** for more details.
`param1`	in	The first parameter of the notification.
`param2`	in	The second parameter of the notification.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35 IDeckLinkEncoderInput Interface

The **IDeckLinkEncoderInput** object interface allows an application to capture an encoded video and audio stream from a DeckLink device.

An **IDeckLinkEncoderInput** interface can be obtained from an **IDeckLink** object interface using QueryInterface. If QueryInterface for an input interface is called on a device which does not support encoded capture, then **QueryInterface** will fail and return E_NOINTERFACE.

Encoded Video capture operates in a push model with encoded video data delivered to an **IDeckLinkEncoderInputCallback** object interface. Audio capture is optional and can be handled by using the same callback object.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	An **IDeckLinkEncoderInput** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkDisplayModeIterator`	`IID_IDeckLinkDisplayModeIterator`	**IDeckLinkEncoderInput::GetDisplayModeIterator** returns an **IDeckLinkDisplayModeIterator** object interface
`IDeckLinkEncoderInputCallback`	`IID_IDeckLinkEncoderInputCallback`	An **IDeckLinkEncoderInputCallback** object interface may be registered with **IDeckLinkEncoderInput::SetCallback**

Public Member Functions	
Method	**Description**
DoesSupportVideoMode	Check whether a given video mode is supported for input
GetDisplayModeIterator	Get an iterator to enumerate the available input display modes
EnableVideoInput	Configure video input
DisableVideoInput	Disable video input
EnableAudioInput	Configure audio input
DisableAudioInput	Disable audio input
StartStreams	Start encoded capture
StopStreams	Stop encoded capture
PauseStreams	Pause encoded capture
FlushStreams	Removes any buffered video and audio frames.
SetCallback	Register input callback
GetHardwareReferenceClock	Get the hardware system clock
SetMemoryAllocator	Register custom memory allocator for encoded video packets
GetAvailableAudioSampleFrameCount	Query audio buffer status – for pull model audio.

2.5.35.1 IDeckLinkEncoderInput::DoesSupportVideoMode

The **DoesSupportVideoMode** method indicates whether a given display mode is supported on encoder input.

Modes may be supported, unsupported or supported with conversion.

Syntax

```
HRESULT     DoesSupportVideoMode
            (BMDDisplayMode displayMode, BMDPixelFormat pixelFormat,
            BMDVideoInputFlags flags, BMDDisplayModeSupport *result,
            IDeckLinkDisplayMode **resultDisplayMode);
```

Parameters

Name	Direction	Description
displayMode	in	Display mode to check
pixelFormat	in	Pixel format to check
flags	in	Video input flags (see **BMDVideoInputFlags** for details).
result	out	Video output mode supported result.
resultDisplayMode	out	If this parameter is not NULL, an **IDeckLinkDisplayMode** object representing the given displayMode is returned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.2 IDeckLinkEncoderInput::GetDisplayModeIterator

The **GetDisplayModeIterator** method returns an iterator which enumerates the available display modes.

Syntax

```
HRESULT     GetDisplayModeIterator
            (IDeckLinkDisplayModeIterator **iterator);
```

Parameters

Name	Direction	Description
iterator	out	display mode iterator

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.3 IDeckLinkEncoderInput::EnableVideoInput

The **EnableVideoInput** method configures video input and puts the hardware into encoded video capture mode. Video input (and optionally audio input) is started by calling **StartStreams**.

Syntax

```
HRESULT      EnableVideoInput
             (BMDDisplayMode displayMode, BMDPixelFormat pixelFormat,
             BMDVideoInputFlags flags);
```

Parameters

Name	Direction	Description
`displayMode`	in	Video mode to capture
`pixelFormat`	in	Encoded pixel format to capture
`flags`	in	Capture flags

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	Is returned on invalid mode or video flags
E_ACCESSDENIED	Unable to access the hardware or input stream currently active
E_OUTOFMEMORY	Unable to create a new frame

2.5.35.4 IDeckLinkEncoderInput::DisableVideoInput

The **DisableVideoInput** method disables the hardware video capture mode.

Syntax

```
HRESULT      DisableVideoInput ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.5 IDeckLinkEncoderInput::EnableAudioInput

The **EnableAudioInput** method configures audio input and puts the hardware into audio capture mode. Encoded audio and video input is started by calling **StartStreams.**

Syntax

```
HRESULT      EnableAudioInput
             (BMDAudioFormat audioFormat,
             BMDAudioSampleRate sampleRate,
             BMDAudioSampleType sampleType, uint32_t channelCount);
```

Parameters

Name	Direction	Description
`audioFormat`	in	Audio format to encode.
`sampleRate`	in	Sample rate to capture
`sampleType`	in	Sample type to capture
`channelCount`	in	Number of audio channels to capture – only 2, 8 or 16 channel capture is supported.

Return Values

Value	Description
E_FAIL	Failure
E_INVALIDARG	Invalid audio format or number of channels requested
E_ACCESSDENIED	Unable to access the hardware or input stream currently active
S_OK	Success

2.5.35.6 IDeckLinkEncoderInput::DisableAudioInput

The **DisableAudioInput** method disables the hardware audio capture mode.

Syntax

```
HRESULT      DisableAudioInput ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.7 IDeckLinkEncoderInput::StartStreams

The **StartStreams** method starts encoded video and audio capture as configured with **EnableVideoInput** and optionally **EnableAudioInput**.

Syntax

```
HRESULT       StartStreams ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	Input stream is already running.
E_UNEXPECTED	Video and Audio inputs are not enabled.

2.5.35.8 IDeckLinkEncoderInput::StopStreams

The **StopStreams** method stops encoded video and audio capture.

Syntax

```
HRESULT       StopStreams ();
```

Parameters

none.

Return Values

Value	Description
E_ACCESSDENIED	Input stream already stopped.
S_OK	Success

2.5.35.9 IDeckLinkEncoderInput::PauseStreams

The **PauseStreams** method pauses encoded video and audio capture. Capture time continues while the streams are paused but no video or audio will be captured. Paused capture may be resumed by calling **PauseStreams** again. Capture may also be resumed by calling **StartStreams** but capture time will be reset.

Syntax

```
HRESULT       PauseStreams ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.10 IDeckLinkEncoderInput::FlushStreams

The **FlushStreams** method removes any buffered video packets and audio frames.

Syntax

```
HRESULT      FlushStreams ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.11 IDeckLinkEncoderInput::SetCallback

The **SetCallback** method configures a callback which will be called as new encoded video, and audio packets become available. Encoder capture is started with **StartStreams**, stopped with **StopStreams** and may be paused with **PauseStreams.**

Syntax

```
HRESULT      SetCallback
             (IDeckLinkEncoderInputCallback *theCallback);
```

Parameters

Name	Direction	Description
`theCallback`	in	Callback object implementing the **IDeckLinkEncoderInputCallback** object interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.12 IDeckLinkEncoderInput::GetHardwareReferenceClock

The **GetHardwareReferenceClock** method returns a clock that is locked to the system clock. The absolute values returned by this method are meaningless, however the relative differences between subsequent calls can be used to determine elapsed time. This method can be called while video input is enabled (see **IDeckLinkEncoderInput::EnableVideoInput** for details).

Syntax

```
HRESULT     GetHardwareReferenceClock
            (BMDTimeScale desiredTimeScale,
            BMDTimeValue *hardwareTime, BMDTimeValue *timeInFrame,
            BMDTimeValue *ticksPerFrame);
```

Parameters

Name	Direction	Description
desiredTimeScale	in	Desired time scale
hardwareTime	out	Hardware reference time (in units of desiredTimeScale)
timeInFrame	out	Time in frame (in units of desiredTimeScale)
ticksPerFrame	out	Number of ticks for a frame (in units of desiredTimeScale)

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.13 IDeckLinkEncoderInput::SetMemoryAllocator

The **SetMemoryAllocator** method sets a custom memory allocator for encoded video packet allocations during capture. Use of a custom memory allocator is optional.

Syntax

```
HRESULT     SetMemoryAllocator
            (IDeckLinkMemoryAllocator *theAllocator);
```

Parameters

Name	Direction	Description
theAllocator	in	Allocator object with an **IDeckLinkMemoryAllocator** interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.35.14 IDeckLinkEncoderInput:: GetAvailableAudioSampleFrameCount

The **GetAvailableAudioSampleFrameCount** method returns the number of audio sample frames currently buffered. Use of this method is only required when using pull model audio – the same audio data is made available via **IDeckLinkEncoderInputCallback** and may be ignored.

Syntax

```
HRESULT     GetAvailableAudioSampleFrameCount
            (uint32_t *availableSampleFrameCount);
```

Parameters

Name	Direction	Description
availableSampleFrameCount	out	The number of buffered audio frames currently available.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.36 IDeckLinkEncoderInputCallback Interface

The **IDeckLinkEncoderInputCallback** object interface is a callback class which is called to provide encoded video packets and audio data during an encoded capture operation.

Related Interfaces

Interface	Interface ID	Description
IDeckLinkEncoder Input	IID_IDeckLinkEncoder Input	An **IDeckLinkEncoderInputCallback** object interface may be registered with **IDeckLinkEncoderInput::SetCallback**
IDeckLinkEncoder VideoPacket	IID_ IDeckLinkEncoder VideoPacket	An **IDeckLinkEncoderVideoPacket** object interface is passed to **IDeckLinkEncoderInputCallback::VideoPacketArrived**
IDeckLinkEncoder AudioPacket	IID_ IDeckLinkEncoder AudioPacket	An **IDeckLinkEncoderAudioPacket** object interface is passed to **IDeckLinkEncoderInputCallback::AudioPacketArrived**

Public Member Functions	
Method	**Description**
VideoInputSignalChanged	Called when a video input signal change is detected
VideoPacketArrived	Called when new video data is available
AudioPacketArrived	Called when new audio data is available

2.5.36.1 IDeckLinkEncoderInputCallback:: VideoInputSignalChanged method

The VideoInputSignalChanged method is called when a video signal change has been detected by the hardware.

To enable this feature, the **bmdVideoInputEnableFormatDetection** flag must be set when calling **IDeckLinkEncoderInput::EnableVideoInput()**.

Syntax

```
HRESULT     VideoInputSignalChanged
            (BMDVideoInputFormatChangedEvents notificationEvents,
            IDeckLinkDisplayMode *newDisplayMode,
            BMDDetectedVideoInputFormatFlags detectedSignalFlags);
```

Parameters

Name	Direction	Description
`notificationEvents`	in	The notification events
`newDisplayMode`	in	The new display mode.
`detectedSignalFlags`	in	The detected signal flags.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.36.2 IDeckLinkEncoderInputCallback::VideoPacketArrived

The **VideoPacketArrived** method is called when an encoded packet has arrived. The method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

When encoded capture is started using **bmdFormatH265**, this callback is used to deliver VCL and non-VCL NAL units.

Syntax

```
HRESULT     VideoPacketArrived
            (IDeckLinkEncoderVideoPacket* videoPacket);
```

Parameters

Name	Direction	Description
`videoPacket`	in	The encoded packet that has arrived. The packet is only valid for the duration of the callback. To hold on to the packet beyond the callback call **AddRef**, and to release the packet when it is no longer required call **Release**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.36.3 IDeckLinkEncoderInputCallback::AudioPacketArrived

The **AudioPacketArrived** method is called when audio capture is enabled with **IDeckLinkEncoderInput::EnableAudioInput**, and an audio packet has arrived. The method is abstract in the base interface and must be implemented by the application developer.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT     AudioPacketArrived
            (IDeckLinkEncoderAudioPacket* audioPacket);
```

Parameters

Name	Direction	Description
audioPacket	in	The audio packet that has arrived. The audio packet is only valid for the duration of the callback. To hold on to the audio packet beyond the callback call **AddRef,** and to release the audio packet when it is no longer required call **Release**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.37 IDeckLinkEncoderPacket Interface

The **IDeckLinkEncoderPacket** object interface represents an encoded data packet.

The **GetSize** method may be used to determine the size of the encoded packet.

Related Interfaces

Interface	Interface ID	Description
IDeckLinkEncoderVideoPacket	IID_IDeckLinkEncoder VideoPacket	**IDeckLinkEncoderVideoPacket** subclasses **IDeckLinkEncoderPacket**
IDeckLinkEncoderAudioPacket	IID_IDeckLinkEncoder AudioPacket	**IDeckLinkEncoderAudioPacket** subclasses **IDeckLinkEncoderPacket**

Public Member Functions	
Method	**Description**
GetBytes	Get pointer to encoded packet data
GetSize	Get size of encoded packet data
GetStreamTime	Get video packet timing information
GetPacketType	Get video packet type

2.5.37.1 IDeckLinkEncoderPacket::GetBytes method

The **GetBytes** method allows direct access to the data buffer of an encoded packet.

Syntax

```
HRESULT      GetBytes (void **buffer);
```

Parameters

Name	Direction	Description
buffer	out	Pointer to raw encoded buffer – only valid while object remains valid.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.37.2 IDeckLinkEncoderPacket::GetSize method

The **GetSize** method returns the number of bytes in the encoded packet.

Syntax

```
long GetSize ();
```

Return Values

Value	Description
BytesCount	Number of bytes in the encoded packet buffer

2.5.37.3 IDeckLinkEncoderPacket::GetStreamTime method

The **GetStreamTime** method returns the time of an encoded video packet for a given timescale.

Syntax

```
HRESULT      GetStreamTime
             (BMDTimeValue *frameTime, BMDTimeScale timeScale);
```

Parameters

Name	Direction	Description
frameTime	out	Frame time (in units of timeScale)
timeScale	in	Time scale for output parameters

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5.37.4 IDeckLinkEncoderPacket::GetPacketType method

The **GetPacketType** method returns the packet type of the encoded packet.

Syntax

```
BMDPacketType       GetPacketType ();
```

Return Values

Value	Description
PacketType	Packet type of encoded packet (**BMDPacketType**)

2.5.38 IDeckLinkEncoderVideoPacket Interface

The **IDeckLinkEncoderVideoPacket** object interface represents an encoded video packet which has been captured by an **IDeckLinkEncoderInput** object interface. **IDeckLinkEncoderVideoPacket** is a subclass of **IDeckLinkEncoderPacket** and inherits all its methods.

The data in the encoded packet is encoded according to the pixel format returned by **GetPixelFormat** – see **BMDPixelFormat** for details.

Objects with an **IDeckLinkEncoderPacket** interface are passed to the **IDeckLinkEncoderInputCallback::VideoPacketArrived** callback.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkEncoderInput`	`IID_IDeckLinkEncoderInput`	Encoded input packets are passed to **IDeckLinkEncoderInputCallback::VideoPacketArrived** by the **IDeckLinkEncoderInput interface**
`IDeckLinkEncoderPacket`	`IID_IDeckLink EncoderPacket`	**IDeckLinkEncoderVideoPacket** subclasses **IDeckLinkEncoderPacket**
`IDeckLinkH265NALPacket`	`IID_IDeckLink H265NALPacket`	**IDeckLinkH265NALPacket** is available from **IDeckLinkEncoderVideoPacket** via QueryInterface

Public Member Functions	
Method	**Description**
GetPixelFormat	Get pixel format for video packet
GetHardwareReferenceTimestamp	Get hardware reference timestamp
GetTimecode	Gets timecode information

2.5.38.1 IDeckLinkEncoderVideoPacket::GetPixelFormat method

The **GetPixelFormat** method returns the pixel format of the encoded packet.

Syntax

```
BMDPixelFormat      GetPixelFormat ();
```

Return Values

Value	Description
PixelFormat	Pixel format of encoded packet **(BMDPixelFormat)**

2.5.38.2 IDeckLinkEncoderVideoPacket:: GetHardwareReferenceTimestamp method

The **GetHardwareReferenceTimestamp** method returns frame time and frame duration for a given timescale.

Syntax

```
HRESULT     GetHardwareReferenceTimestamp
            (BMDTimeScale timeScale, BMDTimeValue *frameTime,
            BMDTimeValue *frameDuration);
```

Parameters

Name	Direction	Description
timeScale	in	The time scale - see **BMDTimeScale** for details.
frameTime	out	The frame time - see **BMDTimeValue** for details.
frameDuration	out	The frame duration - see **BMDTimeValue** for details.

Return Values

Value	Description
E_INVALIDARG	Timescale is not set
S_OK	Success

2.5.38.3 IDeckLinkEncoderVideoPacket::GetTimecode method

The **GetTimecode** method returns the value specified in the ancillary data for the specified timecode type. If the specified timecode type is not found or is invalid, **GetTimecode** returns **S_FALSE**.

Syntax

```
HRESULT      GetTimecode (BMDTimecodeFormat format,
                          IDeckLinkTimecode **timecode);
```

Parameters

Name	Direction	Description
`format`	in	BMDTimecodeFormat to query
`timecode`	out	New **IDeckLinkTimecode** object interface containing the requested timecode or NULL if requested timecode is not available.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_ACCESSDENIED	An invalid or unsupported timecode format was requested.
S_FALSE	The requested timecode format was not present or valid in the ancillary data.

2.5.39 IDeckLinkEncoderAudioPacket Interface

The **IDeckLinkEncoderAudioPacket** object interface represents an encoded audio packet which has been captured by an **IDeckLinkEncoderInput** object interface. **IDeckLinkEncoderAudioPacket** is a subclass of **IDeckLinkEncoderPacket** and inherits all its methods.

The data in the encoded packet is encoded according to the audio format returned by **GetAudioFormat** - see **BMDAudioFormat** for details.

Objects with an **IDeckLinkEncoderAudioPacket** interface are passed to the **IDeckLinkEncoderInputCallback::VideoEncoderAudioPacketArrived** callback.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkEncoderInput`	`IID_IDeckLinkEncoderInput`	Encoded audio packets are passed to **IDeckLinkEncoderInputCallback::AudioPacketArrived** by the **IDeckLinkEncoderInput** interface
`IDeckLinkEncoderPacket`	`IID_IDeckLinkEncoderPacket`	**IDeckLinkEncoderAudioPacket** subclasses **IDeckLinkEncoderPacket**

Public Member Functions	
Method	**Description**
GetAudioFormat	Get audio format for packet

2.5.39.1 IDeckLinkEncoderAudioPacket::GetAudioFormat method

The **GetAudioFormat** method returns the audio format of the encoded packet

Syntax

```
BMDAudioFormat      GetAudioFormat ();
```

Return Values

Value	Description
AudioFormat	Audio format of encoded packet **(BMDAudioFormat)**

2.5.40 IDeckLinkH265NALPacket Interface

The **IDeckLinkH265NALPacket** object interface represents a H.265 encoded packet which has been captured by an **IDeckLinkEncoderVideoPacket** object interface. An **IDeckLinkH265NALPacket** instance can be obtained from **IDeckLinkEncoderVideoPacket** via **QueryInterface** when the captured pixel format is **bmdFormatH265**, otherwise QueryInterface will fail and return E_NOINTERFACE.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkEncoderVideoPacket`	`IID_IDeckLinkEncoder VideoPacket`	**IDeckLinkH265NALPacket** is available from **IDeckLinkEncoderVideoPacket** via QueryInterface

Public Member Functions	
Method	**Description**
GetUnitType	The H.265 NAL unit type
GetBytesNoPrefix	The H.265 encoded buffer without the NAL start code prefix.
GetSizeNoPrefix	The size of the encoded buffer without the NAL start code prefix.

2.5.40.1 IDeckLinkH265NALPacket::GetUnitType method

The **GetUnitType** method returns the H.265 NAL packet unit type.

Syntax

```
HRESULT      GetUnitType (uint8_t *unitType);
```

Parameters

Name	Direction	Description
`unitType`	out	H.265 NAL unit type

Return Values

Value	Description
E_INVALIDARG	If unitType is not provided
S_OK	Success

2.5.40.2 IDeckLinkH265NALPacket::GetBytesNoPrefix method

The **GetBytesNoPrefix** method allows direct access to the data buffer of an encoded packet without the NAL start code prefix.

Syntax

```
HRESULT      GetBytesNoPrefix (void **buffer);
```

Parameters

Name	Direction	Description
`buffer`	out	Pointer to raw encoded buffer without start code prefix – only valid while object remains valid.

Return Values

Value	Description
S_OK	Success

2.5.40.3 IDeckLinkH265NALPacket::GetSizeNoPrefix method

The **GetSizeNoPrefix** method returns the number of bytes in the encoded packet without the NAL start code prefix.

Syntax

```
long         GetSizeNoPrefix ();
```

Return Values

Value	Description
BytesCount	Number of bytes in the encoded packet buffer without the start code prefix

2.5.41 IIDeckLinkEncoderConfiguration Interface

The **IDeckLinkEncoderConfiguration** object interface allows querying and modification of DeckLink encoder configuration parameters.

An **IDeckLinkEncoderConfiguration** object interface can be obtained from the **IDeckLinkEncoderInput** interface using QueryInterface.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkEncoderInput`	`IID_IDeckLinkEncoderInput`	DeckLink encoder input interface

Public Member Functions	
Method	**Description**
SetFlag	Sets a boolean value into the configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID.**
GetFlag	Gets the current boolean value of a setting associated with the given **BMDDeckLinkEncoderConfigurationID**.
SetInt	Sets the current int64_t value into the configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.
GetInt	Gets the current int64_t value of a setting associated with the given **BMDDeckLinkEncoderConfigurationID**.
SetFloat	Sets the current double value into the configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.
GetFloat	Gets the current double value of a setting associated with the given **BMDDeckLinkEncoderConfigurationID.**
SetString	Sets the current string value into the configuration setting with the given **BMDDeckLinkEncoderConfigurationID**.
GetString	Gets the current string value of a setting associated with the given **BMDDeckLinkEncoderConfigurationID**.
GetDecoderConfigurationInfo	Retreive a buffer with decoder configuration information.

2.5.41.1 IDeckLinkEncoderConfiguration::SetFlag method

The **SetFlag** method sets a boolean value into the configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT     SetFlag
            (BMDDeckLinkEncoderConfigurationID cfgID, bool value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The boolean value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID**.

2.5.41.2 IDeckLinkEncoderConfiguration::GetFlag method

The **GetFlag** method gets the current boolean value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT     GetFlag
            (BMDDeckLinkEncoderConfigurationID cfgID, bool *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The boolean value that is set in the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID.**

2.5.41.3 IDeckLinkEncoderConfiguration::SetInt method

The **SetInt** method sets the current int64_t value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT     SetInt
            (BMDDeckLinkEncoderConfigurationID cfgID, int64_t value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The integer value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no integer type configuration setting for this operation corresponding to the given **IDeckLinkEncoderConfiguration.**

2.5.41.4 IDeckLinkEncoderConfiguration::GetInt method

The **GetInt** method gets the current int64_t value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT     GetInt
            (BMDDeckLinkEncoderConfigurationID cfgID, int64_t *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The integer value that is set in the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no integer type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID.**

2.5.41.5 IDeckLinkEncoderConfiguration::SetFloat method

The **SetFloat** method sets the current double value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT     SetFloat
            (BMDDeckLinkEncoderConfigurationID cfgID, double value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The double value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID**.

2.5.41.6 IDeckLinkEncoderConfiguration::GetFloat method

The **GetFloat** method gets the current double value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT     GetFloat
            (BMDDeckLinkEncoderConfigurationID cfgID, double *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The double value that is set in the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID**.

2.5.41.7 IDeckLinkEncoderConfiguration::SetString method

The **SetString** method sets the current string value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT      SetString
             (BMDDeckLinkEncoderConfigurationID cfgID, string value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The string to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID.**

2.5.41.8 IDeckLinkEncoderConfiguration::GetString method

The **GetString** method gets the current string value of a configuration setting associated with the given **BMDDeckLinkEncoderConfigurationID**.

Syntax

```
HRESULT      GetString
             (BMDDeckLinkEncoderConfigurationID cfgID, string *value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	out	The string set in the selected configuration setting. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type configuration setting for this operation corresponding to the given **BMDDeckLinkEncoderConfigurationID**.

2.5.41.9 IDeckLinkEncoderConfiguration::GetBytes method

The **GetBytes** method gets the encoder configuration data in a format represented by the given **BMDDeckLinkEncoderConfigurationID**. To determine the size of the buffer required, call **GetBytes** by initially passing **buffer** as NULL. **GetBytes** will return S_OK and **bufferSize** will be updated to the required size.

Syntax

```
HRESULT      GetBytes (BMDDeckLinkEncoderConfigurationID cfgID,
                       void *buffer, uint32_t *bufferSize);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration data format.
`buffer`	out	The buffer in which to return the configuration data, or NULL to determine the required buffer size.
`bufferSize`	in, out	The size of the provided buffer. Will be updated to the number of bytes returned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no encoder configuration data format corresponding to the given **BMDDeckLinkEncoderConfigurationID**.
E_OUTOFMEMORY	The provided buffer is too small.

2.5.42 IDeckLinkStatus Interface

The **IDeckLinkStatus** object interface allows querying of status information associated with a DeckLink device.

The DeckLink Status ID section lists the status information and associated identifiers that can be queried using this object interface. An **IDeckLinkStatus** object interface can be obtained from an **IDeckLink** object interface using **QueryInterface**.

An application may be notified of changes to status information by subscribing to the **bmdStatusChanged** topic using the **IDeckLinkNotification** interface. See **BMDNotifications** for more information.

For an example demonstrating how status information can be queried and monitored, please see the StatusMonitor sample in the DeckLink SDK.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	An **IDeckLinkStatus** object interface may be obtained from **IDeckLink** using **QueryInterface**

Public Member Functions	
Method	**Description**
GetFlag	Gets the current boolean value of a status associated with the given **BMDDeckLinkStatusID**.
GetInt	Gets the current int64_t value of a status associated with the given **BMDDeckLinkStatusID**.
GetFloat	Gets the current double value of a status associated with the given **BMDDeckLinkStatusID**.
GetString	Gets the current string value of a status associated with the given **BMDDeckLinkStatusID**.
GetBytes	Gets the current byte array value of a status associated with the given **BMDDeckLinkStatusID**.

2.5.42.1 IDeckLinkStatus::GetFlag method

The **GetFlag** method gets the current boolean value of a status associated with the given **BMDDeckLinkStatusID**.

Syntax

```
HRESULT      GetFlag (BMDDeckLinkStatusID statusID, bool *value);
```

Parameters

Name	Direction	Description
`statusID`	in	The **BMDDeckLinkStatusID** of the status information item.
`value`	out	The boolean value corresponding to the statusID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type status corresponding to the given **BMDDeckLinkStatusID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.42.2 IDeckLinkStatus::GetInt method

The **GetInt** method gets the current int64_t value of a status associated with the given **BMDDeckLinkStatusID**.

Syntax

```
HRESULT      GetInt (BMDDeckLinkStatusID statusID, int64_t *value);
```

Parameters

Name	Direction	Description
`statusID`	in	The **BMDDeckLinkStatusID** of the status information item.
`value`	out	The integer value corresponding to the statusID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no integer type status corresponding to the given **BMDDeckLinkStatusID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.42.3 IDeckLinkStatus::GetFloat method

The **GetFloat** method gets the current double value of a status associated with the given **BMDDeckLinkStatusID**.

Syntax

```
HRESULT      GetFloat (BMDDeckLinkStatusID statusID, double *value);
```

Parameters

Name	Direction	Description
`statusID`	in	The **BMDDeckLinkStatusID** of the status information item.
`value`	out	The double value corresponding to the statusID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type status corresponding to the given **BMDDeckLinkStatusID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.42.4 IDeckLinkStatus::GetString method

The **GetString** method gets the current string value of a status associated with the given **BMDDeckLinkStatusID**.

Syntax

```
HRESULT      GetString (BMDDeckLinkStatusIt statusID, string *value);
```

Parameters

Name	Direction	Description
`statusID`	in	The **BMDDeckLinkStatusID** of the status information item.
`value`	out	The string value corresponding to the statusID. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type status corresponding to the given **BMDDeckLinkStatusID**.
E_NOTIMPL	The request is correct however it is not supported by the DeckLink hardware.

2.5.42.5 IDeckLinkStatus::GetBytes method

The **GetBytes** method gets the current byte array value of a status associated with the given **BMDDeckLinkStatusID**.

Note: If the size of the buffer is not sufficient, bufferSize will be updated to the required buffer size.

Syntax

```
HRESULT      GetBytes (BMDDeckLinkStatusID statusID, void *buffer,
                       uint32_t *bufferSize);
```

Parameters

Name	Direction	Description
`statusID`	in	The **BMDDeckLinkStatusID** of the status information item.
`buffer`	out	The buffer in which to return the status data.
`bufferSize`	in, out	The size of the provided buffer. Will be updated to the number of bytes returned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no byte array type status corresponding to the given **BMDDeckLinkStatusID.**

2.5.43.4 IDeckLinkVideoFrameMetadataExtensions::GetString method

The **GetString** method gets the current string value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.

Syntax

```
HRESULT      GetString
             (BMDDeckLinkFrameMetadataID metadataID, string *value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDDeckLinkFrameMetadataID** of the metadata information item.
`value`	out	The string value corresponding to the metadataID. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no string type metadata item corresponding to the given **BMDDeckLinkFrameMetadataID**.

2.5.44 IDeckLinkVideoConversion Interface

The **IDeckLinkVideoConversion** object interface provides the capability to copy an image from a source frame into a destination frame converting between the formats as required. A reference to an **IDeckLinkVideoConversion** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateVideoConversionInstance** on other platforms.

Public Member Functions	
Method	**Description**
ConvertFrame	Copies and converts a source frame into a destination frame.

2.5.44.1 IDeckLinkVideoConversion::ConvertFrame method

The **ConvertFrame** method copies the source frame (srcFrame) to the destination frame (dstFrame). The frame dimension and pixel format of the video frame will be converted if possible. The return value for this method should be checked to ensure that the desired conversion is supported.

The **IDeckLinkVideoFrame** object for the destination frame, with the desired properties, can be created using **IDeckLinkOutput::CreateVideoFrame**. Alternatively the destination frame can be created by subclassing **IDeckLinkVideoFrame** and setting properties directly in the subclassed object.

Syntax

```
HRESULT      ConvertFrame (IDeckLinkVideoFrame* srcFrame,
                           IDeckLinkVideoFrame* dstFrame)
```

Parameters

Name	Direction	Description
`srcFrame`	in	The properties of the source frame
`dstFrame`	in	The properties of the destination frame

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_NOTIMPL	Conversion not currently supported
E_OUTOFMEMORY	The provided buffer is too small. **bufferSize** is updated to the required size.

2.5.43 IDeckLinkVideoFrameMetadataExtensions Interface

The **IDeckLinkVideoFrameMetadataExtensions** object interface allows querying of frame metadata associated with an **IDeckLinkVideoFrame**.

An **IDeckLinkVideoFrameMetadataExtensions** object interface may be obtained from an **IDeckLinkVideoFrame** object interface using **QueryInterface** if the **IDeckLinkVideoFrame** implements this optional interface.

An **IDeckLinkVideoFrame** object interface with the **bmdFrameContainsHDRMetadata** flag may use this interface to query the HDR metadata parameters associated with the video frame.

Related Interfaces

Interface	Interface ID	Description
`IDeckLinkVideoFrame`	`IID_IDeckLinkVideoFrame`	An **IDeckLinkVideoFrameMetadataExtensions** object interface may be obtained from **IDeckLinkVideoFrame** using **QueryInterface**

Public Member Functions	
Method	**Description**
GetInt	Gets the current int64_t value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.
GetFloat	Gets the current double value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.
GetFlag	Gets the current boolean value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.
GetString	Gets the current string value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.

2.5.43.1 IDeckLinkVideoFrameMetadataExtensions::GetInt method

The **GetInt** method gets the current int64_t value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.

Syntax

```
HRESULT     GetInt
            (BMDDeckLinkFrameMetadataID metadataID, int64_t *value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDDeckLinkFrameMetadataID** of the metadata information item.
`value`	out	The integer value corresponding to the metadataID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no integer type metadata item corresponding to the given **BMDDeckLinkFrameMetadataID**.

2.5.43.2 IDeckLinkVideoFrameMetadataExtensions::GetFloat method

The **GetFloat** method gets the current double value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.

Syntax

```
HRESULT     GetFloat
            (BMDDeckLinkFrameMetadataID metadataID, double *value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDDeckLinkFrameMetadataID** of the metadata information item.
`value`	out	The double value corresponding to the metadataID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no float type metadata item corresponding to the given **BMDDeckLinkFrameMetadataID**.

2.5.43.3 IDeckLinkVideoFrameMetadataExtensions::GetFlag method

The **GetFlag** method gets the current boolean value of a metadata item associated with the given **BMDDeckLinkFrameMetadataID**.

Syntax

```
HRESULT     GetFlag
            (BMDDeckLinkFrameMetadataID metadataID, bool* value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDDeckLinkFrameMetadataID** of the metadata information item.
`value`	out	The boolean value corresponding to the metadataID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no flag type metadata item corresponding to the given **BMDDeckLinkFrameMetadataID**.

2.6 Streaming Interface Reference

2.6.1 IBMDStreamingDiscovery Interface

The **IBMDStreamingDiscovery** object interface is used to install or remove the callback for receiving streaming device discovery notifications.

A reference to an **IBMDStreamingDiscovery** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateBMDStreamingDiscoveryInstance** on other platforms.

Public Member Functions	
Method	**Description**
InstallDeviceNotifications	Install device notifications callback
UninstallDeviceNotifications	Remove device notifications callback

2.6.1.1 IBMDStreamingDiscovery::InstallDeviceNotifications method

The **InstallDeviceNotifications** method installs the callback which will be called when a new streaming device becomes available.

Note: Only one callback may be installed at a time.

Syntax

```
HRESULT      InstallDeviceNotifications
             (IBMDStreamingDeviceNotificationCallback* theCallback);
```

Parameters

Name	Direction	Description
`theCallback`	in	Callback object implementing the **IBMDStreamingDeviceNotificationCallback** object interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The callback parameter is invalid.
E_UNEXPECTED	An unexpected internal error has occurred.

2.6.1.2 IBMDStreamingDiscovery:: UninstallDeviceNotifications method

The **UninstallDeviceNotifications** method removes the device notifications callback.

Syntax

```
HRESULT      UninstallDeviceNotifications ();
```

Return Values

Value	Description
S_OK	Success
E_UNEXPECTED	An unexpected internal error has occurred.

2.6.2 IBMDStreamingDeviceNotificationCallback Interface

The **IBMDStreamingDeviceNotificationCallback** object interface is a callback class which is called when a streaming device arrives, is removed or undergoes a mode change.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreamingDiscovery`	`IID_IBMDStreamingDiscovery`	An **IBMDStreamingDeviceNotificationCallback** object interface may be installed with **IBMDStreamingDiscovery::InstallDeviceNotifications**

Public Member Functions	
Method	**Description**
StreamingDeviceArrived	Streaming device arrived
StreamingDeviceRemoved	Streaming device removed
StreamingDeviceModeChanged	Streaming device mode changed

2.6.2.1 IBMDStreamingDeviceNotificationCallback:: StreamingDeviceArrived method

The **StreamingDeviceArrived** method is called when a new streaming device becomes available.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      StreamingDeviceArrived (IDeckLink* device);
```

Parameters

Name	Direction	Description
`device`	in	streaming device

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.2.2 IBMDStreamingDeviceNotificationCallback:: StreamingDeviceRemoved method

The **StreamingDeviceRemoved** method is called when a streaming device is removed.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      StreamingDeviceRemoved (IDeckLink* device);
```

Parameters

Name	Direction	Description
device	in	streaming device

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.2.3 IBMDStreamingDeviceNotificationCallback:: StreamingDeviceModeChanged method

The **StreamingDeviceModeChanged** method is called when a streaming device's mode has changed.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      StreamingDeviceModeChanged
             (IDeckLink* device, BMDStreamingDeviceMode mode);
```

Parameters

Name	Direction	Description
device	in	streaming device
mode	in	new streaming device mode after the mode change occurred

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.3 IBMDStreamingVideoEncodingMode Interface

The **IBMDStreamingVideoEncodingMode** object interface represents a streaming video encoding mode.

The encoding mode encapsulates all the available encoder settings such as video codec settings and audio codec settings. To make changes to encoder settings use the **IBMDStreamingMutableVideoEncodingMode** object interface obtained via the **CreateMutableVideoEncodingMode** method.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreamingVideo EncodingMode PresetIterator`	`IID_IBMDStreaming VideoEncodingMode PresetIterator`	**IBMDStreamingVideoEncodingModePresetIterator::Next** returns an **IBMDStreamingVideoEncodingMode** object interface for each available video encoding mode.
`IBMDStreamingMutable VideoEncodingMode`	`IID_IBMDStreamingMutable VideoEncodingMode`	A mutable subclass of **IBMDStreamingVideoEncodingMode** may be created using **CreateMutableVideoEncodingMode**

Public Member Functions	
Method	**Description**
GetName	Get the name describing the video encoding mode.
GetPresetID	Get the unique ID representing the video encoding mode.
GetSourcePositionX	Get the x coordinate of the origin of the video source rectangle.
GetSourcePositionY	Get the y coordinate of the origin of the video source rectangle.
GetSourceWidth	Get the width of the video source rectangle.
GetSourceHeight	Get the height of the video source rectangle.
GetDestWidth	Get the width of the video destination rectangle.
GetDestHeight	Get the height of the video destination rectangle.
GetFlag	Get the current value of a boolean encoding mode setting.
GetInt	Get the current value of a int64_t encoding mode setting.
GetFloat	Get the current value of a double encoding mode setting.
GetString	Get the current value of a string encoding mode setting.
CreateMutableVideoEncodingMode	Create a mutable copy of the IBMDStreamingVideoEncodingMode object interface.

2.6.3.1 IBMDStreamingVideoEncodingMode::GetName method

The **GetName** method returns a string describing the video encoding mode.

Syntax

```
HRESULT      GetName (string name);
```

Parameters

Name	Direction	Description
name	out	Video encoding name. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_POINTER	The name parameter is invalid.

2.6.3.2 IBMDStreamingVideoEncodingMode::GetPresetID method

The **GetPresetID** method returns the unique ID representing the preset video mode.

Syntax

```
unsigned int       GetPresetID ();
```

Return Values

Value	Description
id	Unique ID of preset video mode.

2.6.3.3 IBMDStreamingVideoEncodingMode:: GetSourcePositionX method

The **GetSourcePositionX** method returns the x coordinate of the origin of the source rectangle used for encoding video.

Syntax

```
unsigned int       GetSourcePositionX ();
```

Return Values

Value	Description
xPosition	The x coordindate in pixels for source rectangle origin.

2.6.3.4 IBMDStreamingVideoEncodingMode:: GetSourcePositionY method

The **GetSourcePositionY** method returns the y coordinate of the origin of the source rectangle used for encoding video.

Syntax

```
unsigned int        GetSourcePositionY ();
```

Return Values

Value	Description
yPosition	The y coordindate in pixels for source rectangle origin.

2.6.3.5 IBMDStreamingVideoEncodingMode:: GetSourceWidth method

The **GetSourceWidth** method returns the width of the source rectangle used for encoding video.

Syntax

```
unsigned int        GetSourceWidth ();
```

Return Values

Value	Description
width	Width in pixels of the source rectangle.

2.6.3.6 IBMDStreamingVideoEncodingMode:: GetSourceHeight method

The **GetSourceHeight** method the height of the source rectangle used for encoding video.

Syntax

```
unsigned int        GetSourceHeight ();
```

Return Values

Value	Description
height	Height in pixels of the source rectangle.

2.6.3.7 IBMDStreamingVideoEncodingMode::GetDestWidth method

The **GetDestWidth** method returns the width of the destination rectangle used when encoding video. If the destination rectangle is different to the source rectangle the video will be scaled when encoding.

Syntax

```
unsigned int        GetDestWidth ();
```

Return Values

Value	Description
width	Width in pixels of the destination rectangle.

2.6.3.8 IBMDStreamingVideoEncodingMode::GetDestHeight method

The **GetDestHeight** method returns the height of the destination rectangle used when encoding video. If the destination rectangle is different to the source rectangle the video will be scaled when encoding.

Syntax

```
unsigned int        GetDestHeight ();
```

Return Values

Value	Description
height	Height in pixels of the destination rectangle.

2.6.3.9 IBMDStreamingVideoEncodingMode::GetFlag method

The **GetFlag** method gets the current value of the boolean configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT      GetFlag
             (BMDStreamingEncodingModePropertyID cfgID, boolean* value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDStreamingEncodingModePropertyID** to get flag value.
`value`	out	The value corresponding to cfgID.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.3.10 IBMDStreamingVideoEncodingMode::GetInt method

The **GetInt** method gets the current value of the int64_t configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT      GetInt
             (BMDStreamingEncodingModePropertyID cfgID, int64_t* value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDStreamingEncodingModePropertyID** to get integer value.
`value`	out	The value corresponding to cfgID.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.3.11 IBMDStreamingVideoEncodingMode::GetFloat method

The **GetFloat** gets the current value of the double configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT      GetFloat
             (BMDStreamingEncodingModePropertyID cfgID, double* value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDStreamingEncodingModePropertyID** to get double value.
`value`	out	The value corresponding to cfgID.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.3.12 IBMDStreamingVideoEncodingMode::GetString method

The **GetString** current value of the string configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT      GetString
             (BMDStreamingEncodingModePropertyID cfgID, string value);
```

Parameters

Name	Direction	Description
`cfgID`	in	**BMDStreamingEncodingModePropertyID** to get string value.
`value`	out	The value corresponding to cfgID. This allocated string must be freed by the caller when no longer required.

Return Values

Value	Description
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.
E_OUTOFMEMORY	Unable to allocate memory for string.

2.6.3.13 IBMDStreamingVideoEncodingMode:: CreateMutableVideoEncodingMode method

The **CreateMutableVideoEncodingMode** method creates a new object interface which is a mutable copy of the **IBMDStreamingVideoEncodingMode** object interface.

IBMDStreamingMutableVideoEncodingMode is a subclass of **IBMDStreamingVideoEncodingMode** and inherits all its methods. It provides additional methods to change settings for the encoding of video and audio streams.

Syntax

```
HRESULT     CreateMutableVideoEncodingMode
            (IBMDStreamingMutableVideoEncodingMode**
            newEncodingMode);
```

Parameters

Name	Direction	Description
`newEncodingMode`	out	A new mutable encoding mode object interface.

Return Values

Value	Description
S_OK	Success
E_POINTER	The newEncodingMode parameter is invalid.
E_OUTOFMEMORY	Unable to allocate memory for new object interface.

2.6.4 IBMDStreamingMutableVideoEncodingMode Interface

The **IBMDStreamingMutableVideoEncodingMode** object interface represents a mutable streaming video encoding mode.

Methods are provided to set video codec settings and audio codec settings. Use this object interface if you wish to perform cropping or scaling of the input video frame, adjust the video or audio bit rate and to change other video or audio codec settings.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreamingVideo EncodingMode`	`IID_IBMDStreamingVideo EncodingMode`	An **IBMDStreamingMutableVideoEncodingMode** object interface may be created from an **IBMDStreamingVideoEncodingMode** interface object using its **CreateMutableVideoEncodingMode** method.

Public Member Functions	
Method	**Description**
SetSourceRect	Set the video source rectangle.
SetDestSize	Set the size of the video destination rectangle.
SetFlag	Set the value for a boolean encoding mode setting.
SetInt	Set the value for an int64_t encoding mode setting.
SetFloat	Set the value for a double encoding mode setting.
SetString	Set the value for a string encoding mode setting.

2.6.4.1 IBMDStreamingMutableVideoEncodingMode:: SetSourceRect method

The **SetSourceRect** method sets the source rectangle used for encoding video.

Cropping of the input video frame can be achieved by using a source rectangle that is different to the input video frame dimensions.

When no source rectangle is set, the source rectangle of the parent **IBMDStreamingVideoEncodingMode** object interface will be used by the encoder.

Syntax

```
HRESULT     SetSourceRect
            (uint32_t posX, uint32_t posY, uint32_t width, uint32_t height);
```

Parameters

Name	Direction	Description
`posX`	in	X coordinate of source rectangle origin.
`posY`	in	Y coordinate of source rectangle origin.
`width`	in	Width of source rectangle.
`height`	in	Height of source rectangle.

Return Values

Value	Description
S_OK	Success

2.6.4.2 IBMDStreamingMutableVideoEncodingMode:: SetDestSize method

The **SetDestSize** method sets the destination rectangle used for encoding video.

When the destination rectangle size is set to a different size to the source rectangle size, scaling will be performed by the encoder.

When no destination rectangle size is set, the source rectangle size of the parent **IBMDStreamingVideoEncodingMode** object interface will be used by the encoder.

Syntax

```
HRESULT      SetDestSize (uint32_t width, uint32_t height);
```

Parameters

Name	Direction	Description
`width`	in	Width of destination rectangle.
`height`	in	Height of destination rectangle.

Return Values

Value	Description
S_OK	Success

2.6.4.3 IBMDStreamingMutableVideoEncodingMode:: SetFlag method

The **SetFlag** method sets a boolean value into the configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT      SetFlag
             (BMDStreamingEncodingModePropertyID cfgID, boolean value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The boolean value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.4.4 IBMDStreamingMutableVideoEncodingMode::SetInt method

The **SetInt** method sets an int64_t value into the configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT     SetInt
            (BMDStreamingEncodingModePropertyID cfgID, int64_t value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The integer value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.4.5 IBMDStreamingMutableVideoEncodingMode:: SetFloat method

The **SetFloat** method sets a double value into the configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT     SetFloat
            (BMDStreamingEncodingModePropertyID cfgID, double value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The double value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.4.6 IBMDStreamingMutableVideoEncodingMode::SetString method

The **SetString** method sets a string value into the configuration setting associated with the given **BMDStreamingEncodingModePropertyID**.

Syntax

```
HRESULT     SetString
            (BMDStreamingEncodingModePropertyID cfgID, string value);
```

Parameters

Name	Direction	Description
`cfgID`	in	The ID of the configuration setting.
`value`	in	The string value to set into the selected configuration setting.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	One or more parameters are invalid.

2.6.5 IBMDStreamingVideoEncodingMode::PresetIteratorInterface

The **IBMDStreamingVideoEncodingModePresetIterator** object interface is used to enumerate the available preset video encoding modes.

A device may have a number of preset encoding modes. These are convenient encoding modes which can be used to encode video and audio into formats suitable for a number of commonly available playback devices.

A reference to an **IBMDStreamingVideoEncodingModePresetIterator** object interface may be obtained from an **IBMDStreamingDeviceInput** object interface using the **GetVideoEncodingModePresetIterator** method.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreaming DeviceInput`	`IID_ IBMDStreaming DeviceInput`	**IBMDStreamingDeviceInput::GetVideoEncodingModePresetIterator** returns an **IBMDStreamingVideoEncodingModePresetIterator** object interface.

Public Member Functions	
Method	**Description**
Next	Returns a pointer to an **IBMDStreamingVideoEncodingMode** object interface for an available preset encoding mode.

2.6.5.1 IBMDStreamingVideoEncodingModePresetIterator:: Next method

The **Next** method returns the next available **IBMDStreamingVideoEncodingMode** object interface.

Syntax

```
HRESULT      Next
             (IBMDStreamingVideoEncodingMode** videoEncodingMode);
```

Parameters

Name	Direction	Description
videoEncodingMode	out	**IBMDStreamingVideoEncodingMode** object interface or NULL when no more video encoding modes are available.

Return Values

Value	Description
S_OK	Success
S_FALSE	No (more) preset encoding modes are available.
E_POINTER	The videoEncodingMode parameter is invalid.

2.6.6 IBMDStreamingDeviceInput Interface

The **IBMDStreamingDeviceInput** object interface represents a physical streaming video encoder device.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	An **IBMDStreamingDeviceInput** object interface may be obtained from IDeckLink using **QueryInterface**.
`IBMDStreaming DeviceNotification Callback`	`IID_IBMDStreaming DeviceNotification Callback`	**IBMDStreamingDeviceNotificationCallback::StreamingDeviceArrived** returns an **IDeckLink** object interface representing a streaming video encoder device

Public Member Functions	
Method	**Description**
DoesSupportVideoInputMode	Indicates whether a video input mode is supported by the device
GetVideoInputModeIterator	Get an iterator to enumerate available video input modes
SetVideoInputMode	Set a display mode as the device's video input mode
GetCurrentDetectedVideoInputMode	Get the current video input mode detected by the device
GetVideoEncodingMode	Get the currently configured video encoding mode
GetVideoEncodingModePresetIterator	Get an iterator to enumerate available video encoding mode presets
DoesSupportVideoEncodingMode	Indicates whether a video encoding mode is supported by the device
SetVideoEncodingMode	Set a video encoding mode as the device's current video encoding mode
StartCapture	Start a video encoding capture
StopCapture	Stop a video encoding capture
SetCallback	Set a callback for receiving new video and audio packets

2.6.6.1 IBMDStreamingDeviceInput:: DoesSupportVideoInputMode method

The **DoesSupportVideoInputMode** method indicates whether a given video input mode is supported on the device.

Syntax

```
HRESULT     DoesSupportVideoInputMode
            (BMDDisplayMode inputMode, boolean* result);
```

Parameters

Name	Direction	Description
`inputMode`	in	**BMDDisplayMode** to test for input support.
`result`	out	Boolean value indicating whether the mode is supported.

Return Values

Value	Description
S_OK	Success
E_POINTER	The result parameter is invalid.
E_INVALIDARG	The inputMode parameter is invalid

2.6.6.2 IBMDStreamingDeviceInput:: GetVideoInputModeIterator method

The **GetVideoInputModeIterator** method returns an iterator which enumerates the available video input modes.

Syntax

```
HRESULT     GetVideoInputModeIterator
            (IDeckLinkDisplayModeIterator** iterator);
```

Parameters

Name	Direction	Description
`iterator`	out	Display mode iterator

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_POINTER	The iterator parameter is invalid.

2.6.6.3 IBMDStreamingDeviceInput::SetVideoInputMode method

The **SetVideoInputMode** method configures the device to use the specified video display mode for input.

Syntax

```
HRESULT      SetVideoInputMode (BMDDisplayMode inputMode);
```

Parameters

Name	Direction	Description
inputMode	in	Display mode to set as the input display mode

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The inputMode parameter is invalid.

2.6.6.4 IBMDStreamingDeviceInput:: GetCurrentDetectedVideoInputMode method

The **GetCurrentDetectedVideoInputMode** method returns the current video input display mode as detected by the device.

Syntax

```
HRESULT      GetCurrentDetectedVideoInputMode
             (BMDDisplayMode* detectedMode);
```

Parameters

Name	Direction	Description
detectedMode	out	Display mode the device detected for video input

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The detectedMode parameter is invalid.

2.6.6.5 IBMDStreamingDeviceInput:: GetVideoEncodingMode method

The **GetVideoEncodingMode** method returns the currently configured video encoding mode.

Syntax

```
HRESULT     GetVideoEncodingMode
            (IBMDStreamingVideoEncodingMode** encodingMode);
```

Parameters

Name	Direction	Description
encodingMode	out	Current video encoding mode

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The encodingMode parameter is invalid.

2.6.6.6 IBMDStreamingDeviceInput:: GetVideoEncodingModePresetIterator method

The **GetVideoEncodingModePresetIterator** method returns an iterator which enumerates the available video encoding mode presets.

Different video display modes may have different encoding mode presets.

Syntax

```
HRESULT     GetVideoEncodingModePresetIterator
            (BMDDisplayMode inputMode,
            IBMDStreamingVideoEncodingModePresetIterator** iterator);
```

Parameters

Name	Direction	Description
inputMode	in	The DisplayMode to iterate encoding mode presets for
iterator	out	Video encoding mode preset iterator

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The iterator parameter is invalid.

2.6.6.7 IBMDStreamingDeviceInput:: DoesSupportVideoEncodingMode method

The **DoesSupportVideoEncodingMode** method indicates whether a given video encoding mode is support by the device for the given input display mode. Modes may be supported, not supported or supported with changes. If a mode is supported with changes, the changed mode will be returned by the **changedEncodingMode** parameter.

Syntax

```
HRESULT     DoesSupportVideoEncodingMode
            (BMDDisplayMode inputMode,
            IBMDStreamingVideoEncodingMode* encodingMode,
            BMDStreamingEncodingSupport* result,
            IBMDStreamingVideoEncodingMode** changedEncodingMode);
```

Parameters

Name	Direction	Description
`inputMode`	in	Display mode to be used with the video encoding mode
`encodingMode`	in	Video encoding mode to be tested for support
`result`	out	Indicates whether the mode is supported, not supported or supported with changes
`changedEncodingMode`	out	Changed encoding mode when the mode is supported with changes

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_POINTER	One or more out parameters are invalid
E_INVALIDARG	The encodingMode parameter is invalid

2.6.6.8 IBMDStreamingDeviceInput::SetVideoEncodingMode method

The **SetVideoEncodingMode** method sets the given video encoding mode as the device's current video encoding mode. It is necessary to set a video encoding mode before calling the **StartCapture** method.

Syntax

```
HRESULT     SetVideoEncodingMode
            (IBMDStreamingVideoEncodingMode* encodingMode);
```

Parameters

Name	Direction	Description
`encodingMode`	in	Video encoding mode to be used by the device.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The encodingMode parameter is invalid

2.6.6.9 IBMDStreamingDeviceInput::StartCapture method

The **StartCapture** method starts a capture on the device using the current video encoding mode.

If a callback implementing the **IBMDStreamingH264InputCallback** object interface has been set by the **SetCallback** method, calls will be made as new compressed video and audio packets are made available by the device.

Syntax

```
HRESULT       StartCapture ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.6.10 IBMDStreamingDeviceInput::StopCapture method

The **StopCapture** method stops a capture if a capture is currently in progress.

Syntax

```
HRESULT       StopCapture ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.6.11 IBMDStreamingDeviceInput::SetCallback method

The **SetCallback** method configures a callback which will be called for new input from the device or when the device input changes.

An object shall be passed implementing the **IBMDStreamingH264InputCallback** object interface as the callback to receive callbacks for new video and audio packets.

An existing callback can be removed by passing NULL in the callback parameter.

Syntax

```
HRESULT      SetCallback (IUnknown* theCallback);
```

Parameters

Name	Direction	Description
`theCallback`	in	callback object implementing the **IUnknown** object interface

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.7 IBMDStreamingH264InputCallback Interface

The **IBMDStreamingH264InputCallback** object interface is a callback class which is called when encoded video and audio packets are available or when the video input to the streaming device changes.

Once a capture has been started with the **IBMDStreamingDeviceInput::StartCapture** method, compressed video and audio packets will become available asynchronously.

This callback object interface can also be used to detect changes to the video input display mode and changes to the video input connector, whether or not a capture is in progress.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreamingDeviceInput`	`IID_IBMDStreaming DeviceInput`	An **IBMDStreamingH264InputCallback** object interface may be installed with **IBMDStreamingDeviceInput::SetCallback**

Public Member Functions	
Method	**Description**
H264NALPacketArrived	Called when a NAL video packet is available
H264AudioPacketArrived	Called when an audio packet is available
MPEG2TSPacketArrived	Called when a transport stream packet is available
H264VideoInputConnectorScanningChanged	Called when the video input connect scanning mode has changed
H264VideoInputConnectorChanged	Called when the video input connect connector has changed
H264VideoInputModeChanged	Called when the video input display mode has changed

2.6.7.1 IBMDStreamingH264InputCallback:: H264NALPacketArrived method

The **H264NALPacketArrived** method is called when an **IBMDStreamingH264NALPacket** becomes available from the streaming device while a capture is in progress.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT     H264NALPacketArrived
            (IBMDStreamingH264NALPacket* nalPacket);
```

Parameters

Name	Direction	Description
`nalPacket`	in	NAL packet containing compressed video.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.7.2 IBMDStreamingH264InputCallback:: H264AudioPacketArrived method

The **H264AudioPacketArrived** method is called when an **IBMDStreamingAudioPacket** becomes available from the streaming device while a capture is in progress.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT     H264AudioPacketArrived
            (IBMDStreamingAudioPacket* audioPacket);
```

Parameters

Name	Direction	Description
`audioPacket`	in	Audio packet containing compressed audio.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.7.3 IBMDStreamingH264InputCallback:: MPEG2TSPacketArrived method

The **MPEG2TSPacketArrived** method is called when an **IBMDStreamingMPEG2TSPacket** becomes available from the streaming device while a capture is in progress.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT     MPEG2TSPacketArrived
            (IBMDStreamingMPEG2TSPacket* tsPacket);
```

Parameters

Name	Direction	Description
`tsPacket`	in	MPEG transport stream packet containing video or audio data.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.7.4 IBMDStreamingH264InputCallback:: H264VideoInputConnectorScanningChanged method

The **H264VideoInputConnectorScanningChanged** method is called when the input connect scanning mode has changed.

This method will be called independently of capture state.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      H264VideoInputConnectorScanningChanged ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.7.5 IBMDStreamingH264InputCallback:: H264VideoInputConnectorChanged method

The **H264VideoInputConnectorChanged** method is called when the streaming device detects a change to the input connector.

This method will be called independently of capture state.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      H264VideoInputConnectorChanged ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.7.6 IBMDStreamingH264InputCallback:: H264VideoInputModeChanged method

The **H264VideoInputModeChanged** method is called when the streaming device detects a change to the video input display mode.

This method will be called independently of capture state.

The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT      H264VideoInputModeChanged ();
```

Parameters

none.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.8 IBMDStreamingH264NALPacket Interface

The **IBMDStreamingH264NALPacket** object interface represents an MPEG-4 AVC/H.264 Network Adaptation Layer (NAL) packet.

Objects with an **IBMDStreamingH264NALPacket** object interface are passed to the **IBMDStreamingH264InputCallback::H264NALPacketArrived** callback.

The MPEG-4 AVC/H.264 NAL packet contains the compressed H.264 video bitstream which can be passed to a suitable H.264 video decoder for decoding and display. For some applications it may be more convenient to process NAL video packets instead of processing video carried in transport stream packets.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreaming H264InputCallback`	`IID_IBMDStreaming H264InputCallback`	New MPEG-4 AVC/H.264 NAL packets are passed to the **IBMDStreamingH264InputCallback::H264NALPacketArrived** callback

Public Member Functions	
Method	**Description**
GetPayloadSize	Get number of bytes in the NAL packet
GetBytes	Get pointer to NAL packet data
GetBytesWithSizePrefix	Get pointer to NAL packet data prefixed by a 32bit size value
GetDisplayTime	Get display time for the NAL packet
GetPacketIndex	Not Implemented

2.6.8.1 IBMDStreamingH264NALPacket::GetPayloadSize method

The **GetPayloadSize** method gets the number of bytes in the NAL packet.

Syntax

```
long        GetPayloadSize ();
```

Return Values

Value	Description
Count	NAL packet size in bytes

2.6.8.2 IBMDStreamingH264NALPacket::GetBytes method

The **GetBytes** method returns a pointer to the data buffer of the NAL packet.

Syntax

```
HRESULT     GetBytes (void** buffer);
```

Parameters

Name	Direction	Description
buffer	out	Pointer to NAL packet data buffer – only valid while object remains valid.

Return Values

Value	Description
S_OK	Success
E_POINTER	The parameter is invalid.

2.6.8.3 IBMDStreamingH264NALPacket:: GetBytesWithSizePrefix method

The **GetBytesWithSizePrefix** method returns a pointer to a data buffer starting with a 32bit unsigned integer containing the size of the NAL packet followed by the data buffer of the NAL packet. This arrangement may be required by some video decoders.

Note: The size of the data buffer returned by **GetBytesWithSizePrefix** is 4 bytes larger than the size of the data buffer returned by GetBytes.

Syntax

```
HRESULT     GetBytesWithSizePrefix (void** buffer);
```

Parameters

Name	Direction	Description
buffer	out	Pointer to NAL packet data buffer prefixed by size value – only valid while object remains

Return Values

Value	Description
S_OK	Success
E_POINTER	The parameter is invalid.

2.6.8.4 IBMDStreamingH264NALPacket::GetDisplayTime method

The **GetDisplayTime method** returns the time at which to display the video contained in the NAL packet.
The display time is in units of the requested time scale.

Syntax

```
HRESULT      GetDisplayTime
             (uint64_t requestedTimeScale, uint64_t* displayTime);
```

Parameters

Name	Direction	Description
`requestedTimeScale`	in	Time scale for the displayTime
`displayTime`	out	Time at which to display the video

Return Values

Value	Description
S_OK	Success
E_POINTER	The displayTime parameter is invalid.

2.6.8.5 IBMDStreamingH264NALPacket::GetPacketIndex method

The **GetPacketIndex** method is not implemented.

2.6.9 IBMDStreamingAudioPacket Interface

The **IBMDStreamingAudioPacket** object interface represents an audio packet.

Objects with an **IBMDStreamingAudioPacket** object interface are passed to the **IBMDStreamingH264InputCallback::H264AudioPacketArrived** callback.

The audio packet can contain compressed audio, such as MPEG-2 AAC audio, which can be passed to a suitable audio decoder for decoding and playback. For some applications it may be more convenient to process audio packets instead of processing audio carried in transport stream packets.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreaming H264InputCallback`	`IID_IBMDStreaming H264InputCallback`	New audio packets are passed to the **IBMDStreamingH264InputCallback::H264AudioPacketArrived** callback

Public Member Functions	
Method	**Description**
GetCodec	Get the codec describing the type of audio in the audio packet
GetPayloadSize	Get number of bytes in the audio packet
GetBytes	Get pointer to audio packet data
GetPlayTime	Get the play time for the audio in the audio packet
GetPacketIndex	Not Implemented

2.6.9.1 IBMDStreamingAudioPacket::GetCodec method

The **GetCodec** method returns the codec describing the audio in the packet.

Syntax

```
BMDStreamingAudioCodec     GetCodec ();
```

Return Values

Value	Description
Codec	The codec for the audio in the packet.

2.6.9.2 IBMDStreamingAudioPacket::GetPayloadSize method

The **GetPayloadSize** method gets the number of bytes in the audio packet.

Syntax

```
long          GetPayloadSize ();
```

Return Values

Value	Description
Count	Audio packet size in bytes.

2.6.9.3 IBMDStreamingAudioPacket::GetBytes method

The **GetBytes** method returns a pointer to the data buffer of the audio packet.

Syntax

```
HRESULT       GetBytes (void** buffer);
```

Parameters

Name	Direction	Description
buffer	out	Pointer to audio packet data buffer – only valid while object remains valid.

Return Values

Value	Description
S_OK	Success
E_POINTER	The parameter is invalid.

2.6.9.4 IBMDStreamingAudioPacket::GetPlayTime method

The **GetPlayTime** method returns the time at which to playback the audio contained in the audio packet.
The play time is in units of the requested time scale.

Syntax

```
HRESULT      GetPlayTime
             (uint64_t requestedTimeScale, uint64_t* playTime);
```

Parameters

Name	Direction	Description
`requestedTimeScale`	in	Time scale for the displayTime
`playTime`	out	Time at which to play the audio

Return Values

Value	Description
S_OK	Success
E_POINTER	The parameter is invalid.

2.6.9.5 IBMDStreamingAudioPacket::GetPacketIndex method

The **GetPacketIndex** method is not implemented.

2.6.10 IBMDStreamingMPEG2TSPacket Interface

The **IBMDStreamingMPEG2TSPacket** object interface represents an MPEG-2 transport stream packet as defined by ISO/IEC 13818-1.

Objects with an **IBMDStreamingMPEG2TSPacket** object interface are passed to the **IBMDStreamingH264InputCallback::MPEG2TSPacketArrived** callback.

The MPEG-2 transport stream packet can contain compressed audio or video together with metadata for decoding and synchronizing audio and video streams. For some applications it may be more convenient to process transport stream packets as an alternative to processing NAL video packets and audio packets separately.

Related Interfaces

Interface	Interface ID	Description
`IBMDStreaming H264InputCallback`	`IID_IBMDStreaming H264InputCallback`	New MPEG-2 transport stream packets are passed to the **IBMDStreamingH264InputCallback::MPEG2TSPacketArrived** callback

Public Member Functions	
Method	**Description**
GetPayloadSize	Get number of bytes in the MPEG-2 transport stream packet
GetBytes	Get pointer to MPEG-2 transport stream packet

2.6.10.1 IBMDStreamingMPEG2TSPacket::GetPayloadSize method

The **GetPayloadSize** method returns the number of bytes in the MPEG-2 transport stream packet including the header.

Syntax

```
long        GetPayloadSize ();
```

Return Values

Value	Description
Count	The size of the MPEG TS packet in bytes.

2.6.10.2 IBMDStreamingMPEG2TSPacket::GetBytes method

The **GetBytes** method returns a pointer to the data buffer of the MPEG-2 transport stream packet.

Syntax

```
HRESULT     GetBytes (void** buffer);
```

Parameters

Name	Direction	Description
`buffer`	out	Pointer to MPEG-2 transport stream packet data buffer - only valid while object remains valid.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_POINTER	The parameter is invalid

2.6.11 IBMDStreamingH264NALParser Interface

The **IBMDStreamingH264NALParser** object interface is used to retrieve video codec settings from a NAL packet.

A reference to an **IBMDStreamingH264NALParser** object interface may be obtained from **CoCreateInstance** on platforms with native COM support or from **CreateBMDStreamingH264NALParser** on other platforms.

Related Interfaces

Interface	Interface ID	Description
`BMDStreamingH264NALPacket`	`IID_IBMDStreamingH264NALPacket`	The NAL packet to be parsed by a method in the **IBMDStreamingH264NALParser** object interface

Public Member Functions	
Method	**Description**
IsNALSequenceParameterSet	Get the packet's Sequence Parameter Set setting
IsNALPictureParameterSet	Get the packet's Picture Parameter Set setting
GetProfileAndLevelFromSPS	Get the packet's profile and level setting

2.6.11.1 IBMDStreamingH264NALParser:: IsNALSequenceParameterSet method

The **IsNALSequenceParameterSet** method parses the specified NAL packet to determine if the Sequence Parameter Set (SPS) decoding parameter has been set in the NAL packet.

Syntax

```
HRESULT      IsNALSequenceParameterSet
             (IBMDStreamingH264NALPacket* nal);
```

Parameters

Name	Direction	Description
`nal`	in	The NAL Packet to query for the state of the sequence parameter.

Return Values

Value	Description
S_OK	The sequence parameter of the NAL packet is set.
S_FALSE	The sequence parameter of the NAL packet is not set.

2.6.11.2 IBMDStreamingH264NALParser:: IsNALPictureParameterSet method

The **IsNALPictureParameterSet** method parses the specified NAL packet to determine if the Picture Parameter Set (PPS) decoding parameter has been set in the NAL packet. This information can be used to configure a decoder for decoding the video contained in the NAL packet.

Syntax

```
HRESULT      IsNALPictureParameterSet
             (IBMDStreamingH264NALPacket* nal);
```

Parameters

Name	Direction	Description
`nal`	in	The NAL Packet to query for the state of the picture parameter.

Return Values

Value	Description
S_OK	The picture parameter of the NAL packet is set.
S_FALSE	The picture parameter of the NAL packet is not set.

2.6.11.3 IBMDStreamingH264NALParser:: GetProfileAndLevelFromSPS method

The **GetProfileAndLevelFromSPS** method parses the specified NAL packet and returns the H.264 profile, level and profile compatibility flags. These values can be used to determine if the video contained in the NAL packet can be decoded by a certain H.264 decoder.

Syntax

```
HRESULT      GetProfileAndLevelFromSPS
             (IBMDStreamingH264NALPacket* nal, uint32_t* profileIdc,
             uint32_t* profileCompatability, uint32_t* levelIdc);
```

Parameters

Name	Direction	Description
`nal`	in	The NAL Packet to query for the profile and level.
`profileIdc`	out	The H.264 profile for this NAL packet.
`profileCompatability`	out	The set of profile constraint flags for this NAL packet.
`levelIdc`	out	The H.264 level for this NAL packet.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_POINTER	One or more parameters are invalid.

2.7 Common Data Types

2.7.1 Basic Types

`boolean`

boolean is represented differently on each platform by using its system type:

Windows	BOOL
macOS	bool
Linux	bool

`Strings`

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

Windows	BSTR
macOS	CFStringRef
Linux	const char *

`int64_t`

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

Windows	LONGLONG
macOS	int64_t
Linux	int64_t

`uint64_t`

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

Windows	ULONGLONG
macOS	uint64_t
Linux	uint64_t

`uint32_t`

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

Windows	unsigned int
macOS	uint32_t
Linux	uint32_t

`int32_t`

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

Windows	int
macOS	int32_t
Linux	int32_t

`uint16_t`

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

Windows	unsigned short
macOS	uint16_t
Linux	uint16_t

`uint8_t`

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

Windows	unsigned char
macOS	uint8_t
Linux	uint8_t

2.7.2 Time Representation

The API uses a flexible scheme to represent time values which can maintain accuracy for any video or audio rate. Time is always represented as a time scale and a time value. The time scale is a unit of ticks per second specified by the API user. Time values are represented as a number of time units since playback or capture began. The API user should choose a time scale value appropriate to the type of video or audio stream being handled. Some examples are provided below:

Stream type	Suggested time scale	Frame time values
24 fps video	24000	0, 1000, 2000, 3000...
23.98 fps video	24000	0, 1001, 2002, 3003...

`BMDTimeScale`

BMDTimeScale is a large integer type which specifies the time scale for a time measurement in ticks per second.

`BMDTimeValue`

BMDTimeValue is a large integer type which represents a time in units of BMDTimeScale.

`BMDTimecodeUserBits`

BMDTimecodeUserBits is a 32-bit unsigned integer representing timecode user bits.

2.7.3 Display Modes

BMDDisplayMode enumerates the video modes supported for output and input.

Mode	Width	Height	Frames per Second	Fields per Frame	Suggested Time Scale	Display Duration
bmdModeNTSC	720	486	30/1.001	2	30000	1001
bmdModeNTSC2398	720	486	30/1.001*	2	24000*	1001
bmdModeNTSCp	720	486	60/1.001	1	60000	1001
bmdModePAL	720	576	25	2	25000	1000
bmdModePALp	720	576	50	1	50000	1000
bmdModeHD720p50	1280	720	50	1	50000	1000
bmdModeHD720p5994	1280	720	60/1.001	1	60000	1001
bmdModeHD720p60	1280	720	60	1	60000	1000
bmdModeHD1080p2398	1920	1080	24/1.001	1	24000	1001
bmdModeHD1080p24	1920	1080	24	1	24000	1000
bmdModeHD1080p25	1920	1080	25	1	25000	1000
bmdModeHD1080p2997	1920	1080	30/1.001	1	30000	1001
bmdModeHD1080p30	1920	1080	30	1	30000	1000
bmdModeHD1080i50	1920	1080	25	2	25000	1000
bmdModeHD1080i5994	1920	1080	30/1.001	2	30000	1001
bmdModeHD1080i6000	1920	1080	30	2	30000	1000
bmdModeHD1080p50	1920	1080	50	1	50000	1000
bmdModeHD1080p5994	1920	1080	60/1.001	1	60000	1001
bmdModeHD1080p6000	1920	1080	60	1	60000	1000
bmdMode2k2398	2048	1556	24/1.001	1	24000	1001
bmdMode2k24	2048	1556	24	1	24000	1000
bmdMode2k25	2048	1556	25	1	25000	1000
bmdMode2kDCI2398	2048	1080	24/1.001	1	24000	1001
bmdMode2kDCI24	2048	1080	24	1	24000	1000
bmdMode2kDCI25	2048	1080	25	1	25000	1000
bmdMode2kDCI2997	2048	1080	30/1.001	1	30000	1001
bmdMode2kDCI30	2048	1080	30	1	30000	1000
bmdMode2kDCI50	2048	1080	50	1	50000	1000
bmdMode2kDCI5994	2048	1080	60/1.001	1	60000	1001
bmdMode2kDCI60	2048	1080	60	1	60000	1000
bmdMode4K2160p2398	3840	2160	24/1.001	1	24000	1001
bmdMode4K2160p24	3840	2160	24	1	24000	1000
bmdMode4K2160p25	3840	2160	25	1	25000	1000
bmdMode4K2160p2997	3840	2160	30/1.001	1	30000	1001

Mode	Width	Height	Frames per Second	Fields per Frame	Suggested Time Scale	Display Duration
bmdMode4K2160p30	3840	2160	30	1	30000	1000
bmdMode4K2160p50	3840	2160	50	1	50000	1000
bmdMode4K2160p5994	3840	2160	60/1.001	1	60000	1001
bmdMode4K2160p60	3840	2160	60	1	60000	1000
bmdMode4kDCI2398	4096	2160	24/1.001	1	24000	1001
bmdMode4kDCI24	4096	2160	24	1	24000	1000
bmdMode4kDCI25	4096	2160	25	1	25000	1000
bmdMode4kDCI2997	4096	2160	30/1.001	1	30000	1000
bmdMode4kDCI30	4096	2160	30	1	30000	1000
bmdMode4kDCI50	4096	2160	50	1	50000	1000
bmdMode4kDCI5994	4096	2160	60/1.001	1	60000	1001
bmdMode4kDCI60	4096	2160	60	1	60000	1000
bmdMode8K4320p2398	7680	4320	24/1.001	1	24000	1001
bmdMode8K4320p24	7680	4320	24	1	24000	1000
bmdMode8K4320p25	7680	4320	25	1	25000	1000
bmdMode8K4320p2997	7680	4320	30/1.001	1	30000	1001
bmdMode8K4320p30	7680	4320	30	1	30000	1000
bmdMode8K4320p50	7680	4320	50	1	50000	1000
bmdMode8K4320p5994	7680	4320	60/1.001	1	60000	1001
bmdMode8K4320p60	7680	4320	60	1	60000	1000
bmdMode8kDCI2398	8192	4320	24/1.001	1	24000	1001
bmdMode8kDCI24	8192	4320	24	1	24000	1000
bmdMode8kDCI25	8192	4320	25	1	25000	1000
bmdMode8kDCI2997	8192	4320	30/1.001	1	30000	1001
bmdMode8kDCI30	8192	4320	30	1	30000	1000
bmdMode8kDCI50	8192	4320	50	1	50000	1000
bmdMode8kDCI5994	8192	4320	60/1.001	1	60000	1001
bmdMode8kDCI60	8192	4320	60	1	60000	1000
bmdModeCintelRAW	4096*	3072*	24*	1	24000	1000
bmdMode CintelCompressedRAW	4096*	3072*	24*	1	24000	1000

Note: bmdModeNTSC2398 mode will be played out on the SDI output with a frame rate of 29.97 frames per second with 3:2 pull down. Some cards may not support all of these modes.

Note: VANC data widths are the same as the display mode width, with the exception of UHD 4K/8K modes (1080 pixels) and DCI 4K/8K modes (2048 pixels).

Note: The width and height for bmdModeCintelRAW and bmdModeCintelCompressedRAW display modes refer to the maximum uncropped frame resolution. Refer to the bmdDeckLinkFrameMetadataCintelImageWidth and bmdDeckLinkFrameMetadataCintelImageHeight Metadata IDs for the actual frame resolutions.

Note: The bmdModeCintelRAW and bmdModeCintelCompressedRAW modes have a nominal frame rate of 24 frames per second. Use the Scanner API to get and set the actual frame rate.

2.7.4 Pixel Formats

BMDPixelFormat enumerates the pixel formats supported for output and input.

bmdFormat8BitYUV : 'UYVY' 4:2:2 Representation

Four 8-bit unsigned components (CCIR 601) are packed into one 32-bit **little-endian** word.

Word																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								B yte 0							
Y' 1								Cr 0								Y' 0								Cb 0							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

int framesize = (Width * 16 / 8) * Height

= rowbytes * Height

In this format, two pixels fit into 32 bits or 4 bytes, so one pixel fits into 16 bits or 2 bytes. For the row bytes calculation, the image width is multiplied by the number of bytes per pixel. For the frame size calculation, the row bytes are simply multiplied by the number of rows in the frame.

bmdFormat10BitYUV : 'v210' 4:2:2 Representation

Twelve 10-bit unsigned components are packed into four 32-bit **little-endian** words.

Word 0																															
Decreasing Address Order																															
Byte 3										Byte 2								Byte 1								Byte 0					
X	X	Cr 0										Y' 0										Cb 0									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Word 1																															
Decreasing Address Order																															
Byte 3										Byte 2								Byte 1								Byte 0					
X	X	Y' 2										Cb 2										Y' 1									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Word 2																															
Decreasing Address Order																															
Byte 3										Byte 2								Byte 1								Byte 0					
X	X	Cb 4										Y'3										Cr 2									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Word 3																															
Decreasing Address Order																															
Byte 3										Byte 2										Byte 1										Byte 0							
X	X	Y' 5										Cr 4										Y' 4									
		9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

int framesize = ((Width + 47) / 48) * 128 * Height

= rowbytes * Height

In this format, each line of video must be aligned on a 128 byte boundary. Six pixels fit into16 bytes so 48 pixels fit in 128 bytes.
For the row bytes calculation the image width is rounded to the nearest 48 pixel boundary and multiplied by 128.
For the frame size calculation the row bytes are simply multiplied by the number of rows in the frame.

bmdFormat8BitARGB : ARGB (or ARGB32) 4:4:4:4 raw

Four 8-bit unsigned components are packed into one 32-bit little-endian word.
Alpha channel is valid.

Word																															
Decreasing Address Order																															
Byte 3										Byte 2										Byte 1										Byte 0							
B										G										R										A							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

int framesize = (Width * 32 / 8) * Height

= rowbytes * Height

In this format, each pixel fits into 32 bits or 4 bytes. For the row bytes calculation the image width is multiplied by the number of bytes per pixel.
For the frame size calculation, the row bytes are simply multiplied by the number of rows in the frame.

bmdFormat8BitBGRA : BGRA (or RGB32) 4:4:4:x raw

Four 8-bit unsigned components are packed into one 32-bit little-endian word.
The alpha channel may be valid.

Word																															
Decreasing Address Order																															
Byte 3										Byte 2										Byte 1										Byte 0							
X										R										G										B							
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

int framesize = (Width * 32 / 8) * Height

= rowbytes * Height

In this format, each pixel fits into 32 bits or 4 bytes. For the row bytes calculation, the image width is multiplied by the number of bytes per pixel. For the frame size calculation, the row bytes are simply multiplied by the number of rows in the frame.

bmdFormat10BitRGB : 'r210' 4:4:4 raw

Three 10-bit unsigned components are packed into one 32-bit big-endian word.

Word																													
Decreasing Address Order																													
Byte 3								Byte 2								Byte 1								Byte 0						
B Lo								G Lo						B Hi		R Lo				G Hi				X	X	R Hi				
7	6	5	4	3	2	1	0	5	4	3	2	1	0	9	8	3	2	1	0	9	8	7	6	x	x	9	8	7	6	5	4

int framesize = ((Width + 63) / 64) * 256 * Height

= rowbytes * Height

In this format each line of video must be aligned a 256 byte boundary. One pixel fits into 4 bytes so 64 pixels fit into 256 bytes.
For the row bytes calculation, the image width is rounded to the nearest 64 pixel boundary and multiplied by 256.
For the frame size calculation, the row bytes are simply multiplied by the number of rows in the frame.

bmdFormat12BitRGB : 'R12B'

Big-endian RGB 12-bit per component with full range (0-4095). Packed as 12-bit per component.

This 12-bit pixel format is compatible with SMPTE 268M Digital Moving-Picture Exchange version 1, Annex C, Method C4 packing.

int framesize = ((Width * 36) / 8) * Height

= rowbytes * Height

In this format, 8 pixels fit into 36 bytes.

Word 0																													
Decreasing Address Order																													
Byte 3								Byte 2								Byte 1								Byte 0						
B0								G0								G0				R0				R0						
7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0

Word 1																													
Decreasing Address Order																													
Byte 3								Byte 2								Byte 1								Byte 0						
B1				G1				G1								R1								R1				B0		
3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8

Word 2																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
G2								G2				R2				R2								B1							
11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4

Word 3																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
G3								R3								R3				B2				B2							
7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0

Word 4																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
G4				R4				R4								B3								B3				G3			
3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8

Word 5																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R5								R5				B4				B4								G4							
11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4

Word 6																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R6								B5								B5				G5				G5							
7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0

Word 7																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R7				B6				B6								G6								G6				R6			
3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8

Word 8																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
B7								B7				G7				G7								R7							
11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4	3	2	1	0	11	10	9	8	7	6	5	4

bmdFormat12BitRGBLE : 'R12L'

Little-endian RGB 12-bit per component with full range (0-4095). Packed as 12-bit per component.

This 12-bit pixel format is compatible with SMPTE 268M Digital Moving-Picture Exchange version 1, Annex C, Method C4 packing.

int framesize = ((Width * 36) / 8) * Height

= rowbytes * Height

In this format, 8 pixels fit into 36 bytes.

Word 0																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R0								G0				R0				G0								B0							
7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0

Word 1																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R1				B0				R1								G1								B1				G1			
3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8

Word 2																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
B1								R2								G2				R2				G2							
11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4

Word 3																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
B2								R3				B2				R3								G3							
7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0

Word 4																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
B3				G3				B3								R4								G4				R4			
3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8

Word 5																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
G4								B4								R5				B4				R5							
11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4

Word 6																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
G5								B5				G5				B5								R6							
7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0

Word 7																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
G6				R6				G6								B6								R7				B6			
3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8

Word 8																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R7								G7								B7				G7				B7							
11	10	9	8	7	6	5	4	7	6	5	4	3	2	1	0	3	2	1	0	11	10	9	8	11	10	9	8	7	6	5	4

bmdFormat10BitRGBXLE : 'R10l' 4:4:4 raw

Three 10-bit unsigned components are packed into one 32-bit little-endian word.

Word																															
Decreasing Address Order																															
Byte 3								Byte 2								Byte 1								Byte 0							
R								R		G						G				B				B						X	X
9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	x	x

int framesize = ((Width + 63) / 64) * 256 * Height

= rowbytes * Height

In this format each line of video must be aligned a 256 byte boundary. One pixel fits into 4 bytes so 64 pixels fit into 256 bytes.
For the row bytes calculation, the image width is rounded to the nearest 64 pixel boundary and multiplied by 256.
For the frame size calculation, the row bytes are simply multiplied by the number of rows in the frame.

bmdFormat10BitRGBX : ‘R10b’ 4:4:4 raw

Three 10-bit unsigned components are packed into one 32-bit big-endian word.

Word																													
Decreasing Address Order																													
Byte 3								Byte 2								Byte 1								Byte 0						
B						X	X	G				B				R		G						R						
5	4	3	2	1	0	x	x	3	2	1	0	9	8	7	6	1	0	9	8	7	6	5	4	9	8	7	6	5	4	3	2

int framesize = ((Width + 63) / 64) * 256 * Height

= rowbytes * Height

In this format each line of video must be aligned a 256 byte boundary. One pixel fits into 4 bytes so 64 pixels fit into 256 bytes.
For the row bytes calculation, the image width is rounded to the nearest 64 pixel boundary and multiplied by 256.
For the frame size calculation, the row bytes are simply multiplied by the number of rows in the frame.

bmdFormatH265 : ‘hev1’

This pixel format represents compressed H.265 encoded video data.

This pixel format is compatible with ITU-T H.265 High Efficiency Video Coding.

bmdFormatDNxHR : ‘AVdh’

This pixel format represents compressed DNxHR encoded video data.

bmdFormat12BitRAWGRBG : ‘r12p’

This pixel format represents 12-bit RAW data with GRBG bayer pattern.

bmdFormat12BitRAWJPEG : ‘r16p’

This pixel format represents 12-bit RAW data arranged in tiles and JPEG compressed.

2.7.5 Field Dominance

BMDFieldDominance enumerates settings applicable to video fields.

bmdUnknownFieldDominance
Indeterminate field dominance.

bmdLowerFieldFirst
The first frame starts with the lower field (the second-from-the-top scan line).

bmdUpperFieldFirst
The first frame starts with the upper field (the top scan line).

bmdProgressiveFrame
A complete frame containing all scan lines.

bmdProgressiveSegmentedFrame
A progressive frame encoded as a PsF
(See **IDeckLinkDisplayMode::GetFieldDominance**for details)

2.7.6 Frame Flags

BMDFrameFlags enumerates a set of flags applicable to a video frame.

bmdFrameFlagDefault
No other flags applicable.

bmdFrameFlagFlipVertical
Frame should be flipped vertically on output

bmdFrameHasNoInputSource
No input source was detected – frame is invalid

bmdFrameContainsHDRMetadata
Frame contains HDR metadata (See **IDeckLinkVideoFrameMetadataExtensions**)

bmdFrameCapturedAsPsF
Frame captured as PsF

bmdFrameContainsCintelMetadata
Frame contains Cintel metadata (see **IDeckLinkVideoFrameMetadataExtensions**)

2.7.7 Video Input Flags

BMDVideoInputFlags enumerates a set of flags applicable to video input.

bmdVideoInputFlagDefault
No other flags applicable

bmdVideoInputEnableFormatDetection
Enable video input mode detection.
(See **IDeckLinkInputCallback::VideoInputFormatChanged** for details)

bmdVideoInputDualStream3D
Set the DeckLink device to capture the 3D mode version of the selected **BMDDisplayMode** display mode.

2.7.8 Video Output Flags

BMDVideoOutputFlags enumerates flags which control the output of video data.

bmdVideoOutputFlagDefault
No flags applicable.

bmdVideoOutputRP188
Output RP188 timecode. If supplied see:
IDeckLinkMutableVideoFrame::SetTimecode

bmdVideoOutputVANC
Output VANC data. If supplied see: **IDeckLinkMutableVideoFrame::SetAncillaryData**

bmdVideoOutputVITC
Output VITC timecode data. If supplied see:
IDeckLinkMutableVideoFrame::SetTimecode

bmdVideoOutputDualStream3D
Set the DeckLink device to output the 3D version of the selected **BMDDisplayMode** display mode.

2.7.9 Output Frame Completion Results Flags

BMDOutputFrameCompletionResult enumerates the possible frame output completion statuses.

bmdOutputFrameCompleted
Frame was displayed normally

bmdOutputFrameDisplayedLate
Frame was displayed late

bmdOutputFrameDropped
Frame was dropped

bmdOutputFrameFlushed
Frame was flushed

Frames are “flushed” when they have been scheduled but are no longer needed due to an action initiated by the API user e.g. a speed or direction change. If frame scheduling falls behind frame output, the hardware will output the least late frame available. When this happens, the frame will receive a completion status of “displayed late”. Frames that are never displayed due to a less late frame being available will receive a completion status of “dropped”.

2.7.10 Frame preview format

BMD3DPreviewFormat enumerates the dual preview formats available for the DeckLink screen preview. The OpenGL based preview format can be set using **IDeckLinkGLScreenPreviewHelper::SetLinkedFramePreviewFormat.**

The DirectX based preview format can be set using **IDeckLinkDX9ScreenPreviewHelper::Set3DPreviewFormat.**

bmd3DPreviewFormatDefault
Preview frames in the default top-bottom format.

bmd3DPreviewFormatLeftOnly
Preview the left eye frame only.

bmd3DPreviewFormatRightOnly
Preview the right eye frame only.

bmd3DPreviewFormatSideBySide
Preview the frames frame in side by side format

bmd3DPreviewFormatTopBottom
Preview the frames in top-bottom format.

2.7.11 Video IO Support

BMDVideoIOSupport enumerates the capture and playback capabilities of a device.

bmdDeviceSupportsCapture
The DeckLink device supports capture operations.

bmdDeviceSupportsPlayback
The DeckLink device supports playback operation.

2.7.12 Video Connection Modes

BMDVideoConnection enumerates the possible video connection interfaces.

bmdVideoConnectionSDI
SDI video connection

bmdVideoConnectionHDMI
HDMI video connection

bmdVideoConnectionOpticalSDI
Optical SDI connection

bmdVideoConnectionComponent
Component video connection

bmdVideoConnectionComposite
Composite video connection

bmdVideoConnectionSVideo
S-Video connection

2.7.13 Link Configuration

BMDLinkConfiguration enumerates the SDI video link configuration on a DeckLink device.

bmdLinkConfigurationSingleLink
A single link video connection. A single video stream uses one connector.

bmdLinkConfigurationDualLink
A dual-link video connection. A single video stream uses two connectors.

bmdLinkConfigurationQuadLink
A quad-link video connection. A single video stream uses four connectors

2.7.14 Audio Sample Rates

BMDAudioSampleRate enumerates the possible audio sample rates.

bmdAudioSampleRate48kHz
48 kHz sample rate

2.7.15 Audio Sample Types

BMDAudioSampleType enumerates the possible audio sample types.

bmdAudioSampleType16bitInteger
16 bit audio sample

bmdAudioSampleType32bitInteger
32 bit audio sample

2.7.16 DeckLink Information ID

BMDDeckLinkAPIInformationID enumerates a set of information details which may be queried (see **IDeckLinkAPIInformation** Interface for details).

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>BMDDeckLinkAPIVersion</td><td>String</td><td>The user viewable API version number.
This allocated string must be freed by the caller when no longer required.</td></tr>
<tr><td>BMDDeckLinkAPIVersion</td><td>Int</td><td>The API version number.
Format:
<table>
<tr><th colspan="4">Word</th></tr>
<tr><th colspan="4">Decreasing Adress Order</th></tr>
<tr><th>Byte 4</th><th>Byte 3</th><th>Byte 2</th><th>Byte 1</th></tr>
<tr><td>Major Version</td><td>Minor Version</td><td>Sub Version</td><td>Extra</td></tr>
</table></td></tr>
</table>

2.7.17 DeckLink Attribute ID

BMDDeckLinkAttributeID enumerates a set of attributes of a DeckLink device which may be queried (see **IDeckLinkAttributes** Interface for details).

Name	Type	Description
BMDDeckLinkSupportsInternalKeying	Flag	True if internal keying is supported on this device.
BMDDeckLinkSupportsExternalKeying	Flag	True if external keying is supported on this device.
BMDDeckLinkSupportsHDKeying	Flag	True if HD keying is supported on this device.
BMDDeckLinkSerialPortDeviceName	String	The operating system name of the RS422 serial port on this device. This allocated string must be freed by the caller when no longer required.
BMDDeckLinkMaximumAudioChannels	Int	The maximum number of embedded audio channels on digital connections supported by this device.
BMDDeckLinkMaximumAnalog AudioInputChannels	Int	The maximum number of input analog audio channels supported by this device.
BMDDeckLinkMaximumAnalog AudioOutputChannels	Int	The maximum number of output analog audio channels supported by this device.
BMDDeckLinkSupportsInputFormatDetection	Flag	True if input format detection is supported on this device.
BMDDeckLinkHasReferenceInput	Flag	True if the DeckLink device has a genlock reference source input connector.
BMDDeckLinkHasSerialPort	Flag	True if device has a serial port.
BMDDeckLinkNumberOfSubDevices	Int	Some DeckLink hardware devices contain multiple independent sub-devices. This attribute will be equal to one for most devices, or two or more on a card with multiple sub-devices (eg DeckLink Duo).
BMDDeckLinkSubDeviceIndex	Int	Some DeckLink hardware devices contain multiple independent sub-devices. This attribute indicates the index of the sub-device, starting from zero.
BMDDeckLinkVideoOutputConnections	Int	The video output connections supported by the hardware (see **BMDVideoConnection** for more details). Multiple video output connections can be active simultaneously.
BMDDeckLinkAudioOutputConnections	Int	The audio output connections supported by the hardware (see **BMDAudioConnection** for more details). Multiple audio output connections can be active simultaneously. Devices with one or more types of analog connection will have the **bmdAudioConnectionAnalog** flag set. Devices with individually selectable XLR/RCA connectors will additionally have the **bmdAudioConnectionAnalogXLR** and **bmdAudioConnectionAnalogRCA** flags set.
BMDDeckLinkVideoInputConnections	Int	The video input connections supported by the hardware (see **BMDVideoConnection** for more details).
BMDDeckLinkAudioInputConnections	Int	The audio input connections supported by the hardware (see **BMDAudioConnection** for more details).

Name	Type	Description
BMDDeckLinkHasAnalogVideoOutputGain	Flag	True if analog video output gain adjustment is supported on this device.
BMDDeckLinkCanOnlyAdjustOverallVideo OutputGain	Flag	True if only the overall video output gain can be adjusted. In this case, only the luma gain can be accessed with the IDeckLinkConfiguration interface, and it controls all three gains (luma, chroma blue and chroma red).
BMDDeckLinkHasVideoInputAntiAliasingFilter	Flag	True if there is an antialising filter on the analog video input of this device.
BMDDeckLinkHasBypass	Flag	True if this device has loop-through bypass function.
BMDDeckLinkVideoInputGainMinimum	Float	The minimum video input gain in dB for this device.
BMDDeckLinkVideoInputGainMaximum	Float	The maximum video input gain in dB for this device.
BMDDeckLinkVideoOutputGainMinimum	Float	The minimum video output gain in dB for this device.
BMDDeckLinkVideoOutputGainMaximum	Float	The maximum video output gain in dB for this device.
BMDDeckLinkSupportsDesktopDisplay	Flag	True if the extended desktop feature is supported on this device and platform.
BMDDeckLinkVideoIOSupport	Int	The capture and/or playback capability of the device. (See **BMDVideoIOSupport** for more information)
BMDDeckLinkSupportsClockTimingAdjustment	Flag	True if this device supports clock timing adjustment (see **bmdDeckLinkConfigClockTimingAdjustment**).
BMDDeckLinkSupportsFullDuplex	Flag	True if the DeckLink device supports capture and playback simultaneously.
BMDDeckLinkPersistentID	Int	A device specific 32 bit unique identifier.
BMDDeckLinkPairedDevicePersistentID	Int	The device specific 32 bit unique identifier of the paired sub-device (If the DeckLink device has paired sub-devices - e.g. the DeckLink Quad 2).
BMDDeckLinkDeviceGroupID	Int	A 32 bit identifier used to group sub-devices belonging to the same DeckLink hardware device. Supported if the sub-device supports BMDDeckLinkPersistentID
BMDDeckLinkTopologicalID	Int	An identifier for DeckLink devices. This feature is supported on a given device if S_OK is returned. The ID will persist across reboots assuming that devices are not disconnected or moved to a different slot.
BMDDeckLinkSupportsFullFrame ReferenceInputTimingOffset	Flag	True if the DeckLink device supports genlock offset adjustment wider than +/- 511 pixels (see **bmdDeckLinkConfigReferenceInputTimingOffset** for more information).
BMDDeckLinkSupportsSMPTELevelAOutput	Flag	True if SMPTE Level A output is supported on this device.
BMDDeckLinkSupportsDualLinkSDI	Flag	True if SDI dual-link is supported on this device.
BMDDeckLinkSupportsQuadLinkSDI	Flag	True if SDI quad-link is supported on this device.
BMDDeckLinkSupportsIdleOutput	Flag	True if this device supports idle output. (see **BMDIdleVideoOutputOperation** for idle output options).
BMDDeckLinkDeckControlConnections	Int	The deck control connections supported by the hardware (see **BMDDeckControlConnection** for more information).
BMDDeckLinkMicrophoneInputGainMinimum	Float	The minimum microphone input gain in dB for this device.
BMDDeckLinkMicrophoneInputGainMaximum	Float	The maximum microphone input gain in dB for this device.

Name	Type	Description
BMDDeckLinkDeviceInterface	Int	The active device interface (see **BMDDeviceInterface** for more information)
BMDDeckLinkHasLTCTimecodeInput	Flag	True if this device has a dedicated LTC input.
BMDDeckLinkVendorName	String	Hardware vendor name. Returned as a static string which must not be freed by the caller.
BMDDeckLinkDisplayName	String	The device's display name. See **IDeckLink::GetDisplayName**.
BMDDeckLinkModeName	String	Hardware Model Name. See **IDeckLink::GetModelName**.
BMDDeckLinkSupportsDuplex ModeConfiguration	Flag	True if the DeckLink device supports duplex mode configuration. See **bmdDeckLinkConfigDuplexMode** and **BMDDuplexMode** for more information.
BMDDeckLinkSupportsHDRMetadata	Flag	True if the device supports transport of HDR metadata.
BMDDeckLinkAudioInputRCAChannelCount	Int	Number of input audio RCA channels supported by this device.
BMDDeckLinkAudioInputXLRChannelCount	Int	Number of input audio XLR channels supported by this device.
BMDDeckLinkAudioOutputRCAChannelCount	Int	Number of output audio RCA channels supported by this device.
BMDDeckLinkAudioOutputXLRChannelCount	Int	Number of output audio XLR channels supported by this device.
BMDDeckLinkDeviceHandle	String	String representing an unique identifier for the device. The format of the string is "RevisionID:PersistentID:TopologicalID".
BMDDeckLinkSupportsColorspaceMetadata	Flag	True if the device supports transport of Colorspace metadata. See **bmdDeckLinkFrameMetadataColorspace** and **BMDColorspace** for more information.

2.7.18 DeckLink Configuration ID

BMDDeckLinkConfigurationID enumerates the set of configuration settings of a DeckLink device which may be queried or set (see **IDeckLinkConfiguration** Interface for details).

Name	Type	Description
bmdDeckLinkConfigOutput1080pAsPsF	Flag	If set, output 1080 or 2K progressive modes as PsF.
bmdDeckLinkConfigCapture1080pAsPsF	Flag	If set, capture 1080 or 2K progressive modes as PsF.
bmdDeckLinkConfigHDMI3DPackingFormat	Int(64)	The 3D packing format setting. See **BMDVideo3DPackingFormat** for more details.
bmdDeckLinkConfigAnalogAudioConsumerLevels	Flag	If set true the analog audio levels are set to maximum gain on audio input and maximum attenuation on audio output. If set false the selected analog input and output gain levels are used.
bmdDeckLinkConfigFieldFlickerRemoval	Flag	Sets field flicker removal when paused functionality. True if enabled.
bmdDeckLinkConfigHD1080p24ToHD1080i5994Conversion	Flag	True if HD 1080p24 to HD 1080i5994 conversion is enabled.
bmdDeckLinkConfig444SDIVideoOutput	Flag	True if 444 video output is enabled.
bmdDeckLinkConfigBlackVideoOutputDuringCapture	Flag	True if black output during capture is enabled. This feature is only supported on legacy DeckLink devices.

Name	Type	Description
bmdDeckLinkConfigReferenceInputTimingOffset	Int(64)	Adjust genlock timing pixel offset. If the device supports wide genlock offset adjustment (see **BMDDeckLinkSupportsFullFrameReferenceInput** TimingOffset attribute) then the supported range is between +/- half the count of total pixels in the video frame. Otherwise the supported range is +/- 511.
bmdDeckLinkConfigCapturePassThroughMode	Int(64)	The capture pass through mode specifies how the monitoring video output is generated while capture is in progress. See **BMDDeckLinkCapturePassthroughMode** for the available modes.
bmdDeckLinkConfigVideoOutputConnection	Int(64)	The output video connection. See BMDVideoConnection for more details. Enabling video output on one connection will enable output on other available output connections which are compatible. The status of active output connection can be queried with this setting. Multiple video output connections can be active simultaneously. When querying the enabled video outputs, the returned integer is a bitmask of BMDVideoConnection where the corresponding bit is set for each active output connection. When setting active video outputs, only one video output connection can be enabled per call, ie, the integer argument must refer to a single video output connection. Enabling multiple output connections simultaneously requires multiple calls.
bmdDeckLinkConfigVideoOutputConversionMode	Int(64)	Settings for video output conversion. The possible output modes are enumerated by **BMDVideoOutputConversionMode**.
bmdDeckLinkConfigAnalogVideoOutputFlags	Int(64)	Settings for analog video output. **BMDAnalogVideoFlags** enumerates the available analog video flags.
bmdDeckLinkConfigVideoInputConnection	Int(64)	The input video connection. Only one video input connection can be active at a time. See **BMDVideoConnection** for more details.
bmdDeckLinkConfigAnalogVideoInputFlags	Int(64)	The analog video input flags. See **BMDAnalogVideoFlags** for more details.
bmdDeckLinkConfigVideoInputConversionMode	Int(64)	The video input conversion mode. See **BMDVideoInputConversionMode** for more details.
bmdDeckLinkConfig32PulldownSequenceInitial TimecodeFrame	Int(64)	The A-frame setting for NTSC 23.98, which is used to appropriately adjust the timecode. The frame setting range is between 0 and 29.
bmdDeckLinkConfigVANCSourceLine1Mapping	Int(64)	The configuration of up to three lines of VANC to be transferred to or from the active picture on capture or output. The acceptable range is between 0 and 30. A value of 0 will disable the capture of that line.
bmdDeckLinkConfigVANCSourceLine2Mapping		The acceptable range is between 0 and 30. A value of 0 will disable the capture of the line.
bmdDeckLinkConfigVANCSourceLine3Mapping		The acceptable range is between 0 and 30. A value of 0 will disable the capture of the line.

Name	Type	Description
bmdDeckLinkConfigAudioInputConnection	Int(64)	The configuration of the audio input connection. See **BMDAudioConnection** for more details.
bmdDeckLinkConfigAnalogAudioInputScaleChannel1 **bmdDeckLinkConfigAnalogAudioInputScaleChannel2** **bmdDeckLinkConfigAnalogAudioInputScaleChannel3** **bmdDeckLinkConfigAnalogAudioInputScaleChannel4**	Float	The analog audio input scale in dB. The supported range is between -12.00 and 12.00.
bmdDeckLinkConfigDigitalAudioInputScale	Float	The digital audio input scale in dB. The acceptable range is between -12.00 and 12.00.
bmdDeckLinkConfigAudioOutputAESAnalogSwitch	Int(64)	The AES / analog audio output selection switch. This is applicable only to cards that support switchable analog audio outputs.
bmdDeckLinkConfigAnalogAudioOutputScaleChannel1 **bmdDeckLinkConfigAnalogAudioOutputScaleChannel2** **bmdDeckLinkConfigAnalogAudioOutputScaleChannel3** **bmdDeckLinkConfigAnalogAudioOutputScaleChannel4**	Float	The analog audio output scale in dB. The acceptable range is between -12.00 and 12.00.
bmdDeckLinkConfigDigitalAudioOutputScale	Float	The digital audio output scale in dB. The acceptable range is between -12.00 and 12.00.
bmdDeckLinkConfigDownConversionOnAllAnalogOutput	Flag	Enable down conversion on all analog outputs.
bmdDeckLinkConfigSMPTELevelAOutput	Flag	Enable SMPTE level A output.
bmdDeckLinkConfigDeviceInformationLabel	string	Set the label of the device. This can only be set if the device has a persistent ID. This information will be saved onto the local machine but not onto the device. This information will also appear in Product Notes section of the Desktop Video Utility.
bmdDeckLinkConfigDeviceInformationSerialNumber	string	Set the serial number of the device. This can only be set if the device has a persistent ID. This information will be saved onto the local machine but not onto the device. This information will also appear in Product Notes section of the Desktop Video Utility.
bmdDeckLinkConfigDeviceInformationCompany	string	Set the device's seller name. This can only be set if the device has a persistent ID. This information will be saved onto the local machine but not onto the device. This information will also appear in Product Notes section of the Desktop Video Utility.
bmdDeckLinkConfigDeviceInformationPhone	string	Set the device's seller phone number. This can only be set if the device has a persistent ID. This information will be saved onto the local machine but not onto the device. This information will also appear in Product Notes section of the Desktop Video Utility.

Name	Type	Description
bmdDeckLinkConfigDeviceInformationEmail	string	Set the device's seller email address. This can only be set if the device has a persistent ID. This information will be saved onto the local machine but not onto the device. This information will also appear in Product Notes section of the Desktop Video Utility.
bmdDeckLinkConfigDeviceInformationDate	string	Set the device's purchase date. This can only be set if the device has a persistent ID. This information will be saved onto the local machine but not onto the device. This information will also appear in Product Notes section of the Desktop Video Utility.
bmdDeckLinkConfigVideoOutputIdleOperation	Int(64)	Video output idle control. See **BMDIdleVideoOutputOperation** for more details.
bmdDeckLinkConfigSwapSerialRxTx	Flag	If set to true, the Rx and Tx lines of the RS422 port on the DeckLink device will be swapped.
bmdDeckLinkConfigBypass	Int(64)	The state of the bypass feature. This parameter can be set to a value of -1 for normal operation or zero to bypass the card. A timeout of up to 65 seconds may be specified in milliseconds. If the timeout is reached without the parameter being reset, the card will be bypassed automatically. The actual timeout will be approximately the time requested.
bmdDeckLinkConfigClockTimingAdjustment	Int(64)	Clock frequency adjustment for fine output control. The acceptable range is from -127 to 127 PPM (Parts Per Million).
bmdDeckLinkConfigVideoInputScanning	Flag	The video input connector scanning on the H.264 Pro Recorder. True if enabled.
bmdDeckLinkConfigUseDedicatedLTCInput	Flag	Use the timecode from the LTC input rather than from the SDI stream.
bmdDeckLinkConfigDefaultVideoOutputMode	Int(64)	The default video output mode. The **bmdDeckLinkConfigDefaultVideoOutputModeFlags** must be set for 3D video modes before using this setting. See BMDDisplayMode for more details.
bmdDeckLinkConfigDefaultVideoOutput ModeFlags	Int(64)	The default video output mode 2D or 3D flag setting. See **bmdVideoOutputFlagDefault** and **bmdVideoOutputDualStream3D** for more details.
bmdDeckLinkConfigSDIOutputLink Configuration	Int(64)	The SDI link configuration for a single output video stream. See **BMDLinkConfiguration** for more information.
bmdDeckLinkConfigVideoOutputComponent LumaGain	Float	The component video output luma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.

Name	Type	Description
bmdDeckLinkConfigVideoOutputComponent ChromaBlueGain	Float	The component video output chroma blue gain in dB.The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoOutputComponent ChromaRedGain	Float	The component video output chroma red gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoOutputComposite LumaGain	Float	The composite video output luma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoOutputComposite ChromaGain	Float	The composite video output chroma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoOutputSVideo LumaGain	Float	The s-video output luma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoOutputSVideo ChromaGain	Float	The s-video output chroma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoOutputGainMinimum** and **BMDDeckLinkVideoOutputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoInputComponent LumaGain	Float	The component video input luma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoInputComponent ChromaBlueGain	Float	The component video input chroma blue gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoInputComponent ChromaRedGain	Float	The component video input chroma red gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoInputComposite LumaGain	Float	The composite video input luma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.

Name	Type	Description
bmdDeckLinkConfigVideoInputComposite ChromaGain	Float	The composite video input chroma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoInputSVideo LumaGain	Float	The s-video input luma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigVideoInputSVideo ChromaGain	Float	The s-video input chroma gain in dB. The accepted range can be determined by using the **BMDDeckLinkVideoInputGainMinimum** and **BMDDeckLinkVideoInputGainMaximum** attributes with **IDeckLinkAttributes** interface.
bmdDeckLinkConfigMicrophone PhantomPower	Flag	If set to true, the Microphone input will provide +48V Phantom Power.
bmdDeckLinkConfigMicrophoneInputGain	Float	The microphone input gain in dB. The acceptable range can be determined via **BMDDeckLinkMicrophoneInputGainMinimum** and **BMDDeckLinkMicrophoneInputGainMaximum**. If set to 0dB, the microphone input will be muted.
bmdDeckLinkConfigHeadphoneVolume	Float	Set the headphone volume, acceptable range is between 0.0 (mute), to 1.0 (full volume)
bmdDeckLinkConfigDeckControlConnection	Int(64)	The active RS422 deck control connection. See **BMDDeckControlConnection** for more information.
bmdDeckLinkConfigSDIInput3D PayloadOverride	Flag	If set to true, the device will capture two genlocked SDI streams with matching video modes as a 3D stream.
bmdDeckLinkConfigDuplexMode	Int(64)	The active duplex mode on sub-devices which support duplex mode configuration. See **BMDDeckLinkSupportsDuplexModeConfiguration** and BMDDuplexMode for more information.
bmdDeckLinkConfigRec2020Output	Flag	If set to true, device will output Rec.709 frames in Rec.2020 colorspace (See **BMDColorspace**)
bmdDeckLinkConfigQuadLinkSDIVideoOutput SquareDivisionSplit	Flag	If set to true, Quad-link SDI is output in Square Division Quad Split mode.

2.7.19 Audio Output Stream Type

BMDAudioOutputStreamType enumerates the Audio output stream type (see **IDeckLinkOutput::EnableAudioOutput** for details).

bmdAudioOutputStreamContinuous
Audio stream is continuous.

bmdAudioOutputStreamContinuousDontResample
Lock audio sample rate. (not currently supported)

bmdAudioOutputStreamTimestamped
Audio stream is time stamped.

2.7.20 Analog Video Flags

BMDAnalogVideoFlags enumerates a set of flags applicable to analog video.

bmdAnalogVideoFlagCompositeSetup75
This flag is only applicable to NTSC composite video and sets the black level to 7.5 IRE, which is used in the USA, rather than the default of 0.0 IRE which is used in Japan.

bmdAnalogVideoFlagComponentBetacamLevels
This flag is only applicable to the component analog video levels. It sets the levels of the color difference channels in accordance to the SMPTE standard or boosts them by a factor of 4/3 for the Betacam format.

2.7.21 Audio Connection Modes

BMDAudioConnection enumerates the possible audio connection interfaces.

bmdAudioConnectionEmbedded
Embedded SDI or HDMI audio connection

bmdAudioConnectionAESEBU
AES/EBU audio connection

bmdAudioConnectionAnalog
Analog audio connection

bmdAudioConnectionAnalogXLR
Analog XLR audio connection

bmdAudioConnectionAnalogRCA
Analog RCA audio connection

bmdAudioConnectionMicrophone
Analog Microphone audio connection

bmdAudioConnectionHeadphones
Analog Headphone audio connection

2.7.22 Audio Output Selection switch

BMDAudioOutputAnalogAESSwitch enumerates the settings of the audio output Analog / AES switch.
Refer to the IDeckLinkConfiguration interface to get and set analog / AES switch settings.

bmdAudioOutputSwitchAESEBU
AES / EBU audio output.

bmdAudioOutputSwitchAnalog
Analog audio output.

2.7.23 Output Conversion Modes

BMDVideoOutputConversionMode enumerates the possible video output conversions.

bmdNoVideoOutputConversion
No video output conversion

bmdVideoOutputLetterboxDownconversion
Down-converted letterbox SD output

bmdVideoOutputAnamorphicDownconversion
Down-converted anamorphic SD output

bmdVideoOutputHD720toHD1080Conversion
HD720 to HD1080 conversion output

bmdVideoOutputHardwareLetterboxDownconversion
Simultaneous output of HD and down-converted letterbox SD

bmdVideoOutputHardwareAnamorphicDownconversion
Simultaneous output of HD and down-converted anamorphic SD

bmdVideoOutputHardwareCenterCutDownconversion
Simultaneous output of HD and center cut SD

bmdVideoOutputHardware720p1080pCrossconversion
The simultaneous output of 720p and 1080p cross-conversion

bmdVideoOutputHardwareAnamorphic720pUpconversion
The simultaneous output of SD and up-converted anamorphic 720p

bmdVideoOutputHardwareAnamorphic1080iUpconversion
The simultaneous output of SD and up-converted anamorphic 1080i

bmdVideoOutputHardwareAnamorphic149To720pUpconversion
The simultaneous output of SD and up-converted anamorphic widescreen aspect ratio 14:9 to 720p.

bmdVideoOutputHardwareAnamorphic149To1080iUpconversion
The simultaneous output of SD and up-converted anamorphic widescreen aspect ratio 14:9 to 1080i.

bmdVideoOutputHardwarePillarbox720pUpconversion
The simultaneous output of SD and up-converted pillarbox 720p

bmdVideoOutputHardwarePillarbox1080iUpconversion
The simultaneous output of SD and up-converted pillarbox 1080i

2.7.24 Input Conversion Modes

BMDVideoInputConversionMode enumerates the possible video input conversions.

bmdNoVideoInputConversion
No video input conversion

bmdVideoInputLetterboxDownconversionFromHD1080
HD1080 to SD video input down conversion

bmdVideoInputAnamorphicDownconversionFromHD1080
Anamorphic from HD1080 to SD video input down conversion

bmdVideoInputLetterboxDownconversionFromHD720
Letter box from HD720 to SD video input down conversion

bmdVideoInputAnamorphicDownconversionFromHD720
Anamorphic from HD720 to SD video input down conversion

bmdVideoInputLetterboxUpconversion
Letterbox video input up conversion

bmdVideoInputAnamorphicUpconversion
Anamorphic video input up conversion

2.7.25 Video Input Format Changed Events

BMDVideoInputFormatChangedEvents enumerates the properties of the video input signal format that have changed.
(See **IDeckLinkInputCallback::VideoInputFormatChanged** for details).

bmdVideoInputDisplayModeChanged
Video input display mode has changed (see **BMDDisplayMode** for details)

bmdVideoInputFieldDominanceChanged
Video input field dominance has changed (see **BMDFieldDominance** for details)

bmdVideoInputColorspaceChanged
Video input color space has changed (see **BMDDetectedVideoInputFormatFlags** for details)

2.7.26 Detected Video Input Format Flags

BMDDetectedVideoInputFormatFlags enumerates the video input signal(See IDeckLinkInputCallback::VideoInputFormatChanged for details)

bmdDetectedVideoInputYCbCr422
The video input detected is YCbCr 4:2:2 represention.

bmdDetectedVideoInputRGB444
The video input detected is RGB 4:4:4 represention.

bmdDetectedVideoInputDualStream3D
The video input detected is dual stream 3D video.

2.7.27 Capture Pass Through Mode

BMDDeckLinkCapturePassthroughMode enumerates whether the video output is electrical connected to the video input or if the clean switching mode is enabled.

bmdDeckLinkCapturePassthroughModeDirect
In direct mode the monitoring video output is directly electrically connected to the video input.

bmdDeckLinkCapturePassthroughModeCleanSwitch
In clean switch mode, the captured video is played back out the monitoring outputs allowing a clean switch between monitoring and playback if the video modes are compatible. The monitoring output signal is affected by the options specified on capture and some latency is introduced between capture and monitoring.

bmdDeckLinkCapturePassthroughModeDisabled
In disabled mode the video input is not displayed out the monitoring outputs, which instead display black frames or the last frame played, dependant on the configuration of the Idle Output setting (see **BMDIdleVideoOutputOperation**).

2.7.28 Display Mode Characteristics

BMDDisplayModeFlags enumerates the possible characteristics of an **IDeckLinkDisplayMode** object.

bmdDisplayModeSupports3D
The 3D equivalent of this display mode is supported by the installed DeckLink device.

bmdDisplayModeColorspaceRec601
This display mode uses the Rec. 601 standard for encoding interlaced analogue video signals in digital form.

bmdDisplayModeColorspaceRec709
This display mode uses the Rec. 709 standard for encoding high definition video content.

bmdDisplayModeColorspaceRec2020
This display mode uses the Rec. 2020 standard for encoding ultra-high definition video content.

2.7.29 Video 3D packing format

The **BMDVideo3DPackingFormat** enumerates standard modes where two frames are packed into one.

bmdVideo3DPackingSidebySideHalf
Frames are packed side-by-side as a single stream.

bmdVideo3DPackingLinebyLine
The two eye frames are packed on alternating lines of the source frame.

bmdVideo3DPackingTopAndBottom
The two eye frames are packed into the top and bottom half of the source frame.

bmdVideo3DPackingFramePacking
Frame packing is a standard HDMI 1.4a 3D mode (Top / Bottom full).

bmdVideo3DPackingLeftOnly
Only the left eye frame is displayed.

bmdVideo3DPackingRightOnly
Only the right eye frame is displayed.

2.7.30 Display Mode Support

BMDDisplayModeSupport enumerates the possible display mode support types.

bmdDisplayModeNotSupported
Display mode is not supported

bmdDisplayModeSupported
Display mode is supported natively

bmdDisplayModeSupportedWithConversion
Display mode is supported with conversion

2.7.31 BMDTimecodeFormat

BMDTimecodeFormat enumerates the possible video frame timecode formats.

bmdTimecodeRP188VITC1
RP188 VITC1 timecode (DBB1=1) on line 9.

bmdTimecodeRP188VITC2
RP188 VITC2 timecode (DBB1=2) on line 571.

bmdTimecodeRP188LTC
RP188 LTC timecode (DBB1=0) on line 10, or the dedicated LTC input if bmdDeckLinkConfigUseDedicatedLTCInput is true.

bmdTimecodeRP188Any
In capture mode the first valid RP188 timecode will be returned. In playback mode the timecode is set as RP188 VITC1.

bmdTimecodeVITC
VITC timecode field 1.

bmdTimecodeVITCField2
VITC timecode field 2.

bmdTimecodeSerial
Serial timecode.

2.7.32 BMDTimecodeFlags

BMDTimecodeFlags enumerates the possible flags that accompany a timecode.

bmdTimecodeFlagDefault
timecode is a non-drop timecode

bmdTimecodeIsDropFrame
timecode is a drop timecode

bmdTimecodeFieldMark
timecode field mark flag used with frame rates above 30 FPS

bmdTimecodeColorFrame
timecode color frame flag

2.7.33 BMDTimecodeBCD

Each four bits represent a single decimal digit:

digit	bit 3	bit 2	bit 1	bit 0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

Word																													
Decreasing Address Order																													
Byte 4								Byte 3								Byte 2								Byte 1					
Tens of hours				hours				Tens of minutes				minutes				Tens of seconds				seconds				Tens of frames				frames			
7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0

2.7.34 Deck Control Mode

BMDDeckControlMode enumerates the possible deck control modes.

bmdDeckControlNotOpened
Deck control is not opened

bmdDeckControlVTRControlMode
Deck control VTR control mode

bmdDeckControlExportMode
Deck control export mode

bmdDeckControlCaptureMode
Deck control capture mode

2.7.35 Deck Control Event

BMDDeckControlEvent enumerates the possible deck control events.

bmdDeckControlAbortedEvent
This event is triggered when a capture or edit-to-tape operation is aborted.

bmdDeckControlPrepareForExportEvent
This export-to-tape event is triggered a few frames before reaching the in-point. At this stage, **IDeckLinkOutput::StartScheduledPlayback()** must be called.

bmdDeckControlExportCompleteEvent
This export-to-tape event is triggered a few frames after reaching the out-point. At this point, it is safe to stop playback. Upon reception of this event the deck's control mode is set back to **bmdDeckControlVTRControlMode**.

bmdDeckControlPrepareForCaptureEvent
This capture event is triggered a few frames before reaching the in-point. The serial timecode attached to **IDeckLinkVideoInputFrames** is now valid.

bmdDeckControlCaptureCompleteEvent
This capture event is triggered a few frames after reaching the out-point. Upon reception of this event the deck's control mode is set back to **bmdDeckControlVTRControlMode**.

2.7.36 Deck Control VTR Control States

BMDDeckControlVTRControlState enumerates the possible deck control VTR control states.

bmdDeckControlNotInVTRControlMode
The deck is currently not in VTR control mode.

bmdDeckControlVTRControlPlaying
The deck is currently playing.

bmdDeckControlVTRControlRecording
The deck is currently recording.

bmdDeckControlVTRControlStill
The deck is currently paused.

bmdDeckControlVTRControlShuttleForward
The deck is currently in shuttle forward mode.

bmdDeckControlVTRControlShuttleReverse
The deck is currently in shuttle reverse mode.

bmdDeckControlVTRControlJogForward
The deck is currently in jog (one frame at a time) forward mode.

bmdDeckControlVTRControlJogReverse
The deck is currently in jog (one frame at a time) reverse mode.

bmdDeckControlVTRControlStopped
The deck is currently stopped.

2.7.37 Deck Control Status Flags

BMDDeckControlStatusFlags enumerates the possible deck control status flags.

bmdDeckControlStatusDeckConnected
The deck has been connected (TRUE) / disconnected (FALSE).

bmdDeckControlStatusRemoteMode
The deck is in remote (TRUE) / local mode (FALSE).

bmdDeckControlStatusRecordInhibited
Recording is inhibited (TRUE) / allowed(FALSE).

bmdDeckControlStatusCassetteOut
The deck does not have a cassette (TRUE).

2.7.38 Deck Control Export Mode Ops Flags

BMDDeckControlExportModeOpsFlags enumerates the possible deck control edit-to-tape and export-to-tape mode operations.

bmdDeckControlExportModeInsertVideo
Insert video

bmdDeckControlExportModeInsertAudio1
Insert audio track 1

bmdDeckControlExportModeInsertAudio2
Insert audio track 2

bmdDeckControlExportModeInsertAudio3
Insert audio track 3

bmdDeckControlExportModeInsertAudio4
Insert audio track 4

bmdDeckControlExportModeInsertAudio5
Insert audio track 5

bmdDeckControlExportModeInsertAudio6
Insert audio track 6

bmdDeckControlExportModeInsertAudio7
Insert audio track 7

bmdDeckControlExportModeInsertAudio8
Insert audio track 8

bmdDeckControlExportModeInsertAudio9
Insert audio track 9

bmdDeckControlExportModeInsertAudio10
Insert audio track 10

bmdDeckControlExportModeInsertAudio11
Insert audio track 11

bmdDeckControlExportModeInsertAudio12
Insert audio track 12

bmdDeckControlExportModeInsertTimeCode
Insert timecode

bmdDeckControlExportModeInsertAssemble
Enable assemble editing.

bmdDeckControlExportModeInsertPreview
Enable preview auto editing

bmdDeckControlUseManualExport
Use edit on/off (TRUE) or autoedit (FALSE). Edit on/off is currently not supported.

2.7.39 Deck Control error

BMDDeckControlError enumerates the possible deck control errors.

bmdDeckControlNoError

bmdDeckControlModeError
The deck is not in the correct mode for the desired operation.
Eg. A play command is issued, but the current mode is not VTRControlMode

bmdDeckControlMissedInPointError
The in point was missed while prerolling as the current timecode has passed the begin in / capture timecode.

bmdDeckControlDeckTimeoutError
Deck control timeout error.

bmdDeckControlCommandFailedError
A deck control command request has failed.

bmdDeckControlDeviceAlreadyOpenedError
The deck control device is already open.

bmdDeckControlFailedToOpenDeviceError
Deck control failed to open the serial device.

bmdDeckControlInLocalModeError
The deck in local mode and is no longer controllable.

bmdDeckControlEndOfTapeError
Deck control has reached or is trying to move past the end of the tape.

bmdDeckControlUserAbortError
Abort an export-to-tape or capture operation.

bmdDeckControlNoTapeInDeckError
There is currently no tape in the deck.

bmdDeckControlNoVideoFromCardError
A capture or export operation was attempted when the input signal was invalid.

bmdDeckControlNoCommunicationError
The deck is not responding to requests.

bmdDeckControlBufferTooSmallError
When sending a custom command, either the internal buffer is too small for the provided custom command (reduce the size of the custom command), or the buffer provided for the command's response is too small (provide a larger one).

bmdDeckControlBadChecksumError
When sending a custom command, the deck's response contained an invalid checksum.

bmdDeckControlUnknownError
Deck control unknown error

2.7.40 Genlock reference status

BMDReferenceStatus enumerates the genlock reference statuses of the DeckLink device.

bmdReferenceNotSupportedByHardware
The DeckLink device does not have a genlock input connector.

bmdReferenceLocked
Genlock reference lock has been achieved.

2.7.41 Idle Video Output Operation

BMDIdleVideoOutputOperation enumerates the possible output modes when idle.

bmdIdleVideoOutputBlack
When not playing video, the device will output black frames.

bmdIdleVideoOutputLastFrame
When not playing video, the device will output the last frame played.

2.7.42 Device Busy State

BMDDeviceBusyState enumerates the possible busy states for a device.

bmdDeviceCaptureBusy
The device is currently being used for capture.

bmdDevicePlaybackBusy
The device is currently being used for playback.

bmdDeviceSerialPortBusy
The device's serial port is currently being used.

2.7.43 DeckLink Device Notification

BMDNotifications enumerates the possible notifications for DeckLink devices.

bmdPreferencesChanged
The preferences have changed. This occurs when **IDeckLinkConfiguration::WriteToPreferences** is called, or when the preference settings are saved in the Blackmagic Design Control Panel. The param1 and param2 parameters are 0.

bmdStatusChanged
A status information item has changed. The param1 parameter contains the **BMDDeckLinkStatusID** of the status information item which changed; param2 is 0. Use the **IDeckLinkStatus** interface to retrieve the new status.

2.7.44 Streaming Device Mode

BMDStreamingDeviceMode enumerates the possible device modes for the streaming device.

bmdStreamingDeviceNotPowered
The streaming device is not powered.

bmdStreamingDeviceBooting
The streaming device is booting.

bmdStreamingDeviceNeedsFirmwareUpdate
The streaming device needs a firmware update.

bmdStreamingDeviceUpdatingFirmware
The streaming device is updating firmware.

bmdStreamingDeviceIdle
The streaming device is idle.

bmdStreamingDeviceEncoding
The streaming device is encoding.

bmdStreamingDeviceStopping
The streaming device is stopping.

bmdStreamingDeviceUnknown
The streaming device is in an unknown state.

2.7.45 Streaming Device Encoding Frame Rates

BMDStreamingEncodingFrameRate enumerates the possible encoded frame rates of the streaming device.

bmdStreamingEncodedFrameRate50i
The encoded interlaced frame rate is 50 fields per second.

bmdStreamingEncodedFrameRate5994i
The encoded interlaced frame rate is 59.94 fields per second.

bmdStreamingEncodedFrameRate60i
The encoded interlaced frame rate is 60 fields per second.

bmdStreamingEncodedFrameRate2398p
The encoded progressive frame rate is 23.98 frames per second.

bmdStreamingEncodedFrameRate24p
The encoded progressive frame rate is 24 frames per second.

bmdStreamingEncodedFrameRate25
The encoded progressive frame rate is 25 frames per second.

bmdStreamingEncodedFrameRate2997p
The encoded progressive frame rate is 29.97 frames per second.

bmdStreamingEncodedFrameRate30p
The encoded progressive frame rate is 30 frames per second.

bmdStreamingEncodedFrameRate50p
The encoded progressive frame rate is 50 frames per second.

bmdStreamingEncodedFrameRate5994p
The encoded progressive frame rate is 59.94 frames per second.

bmdStreamingEncodedFrameRate60p
The encoded progressive frame rate is 60 frames per second.

2.7.46 Streaming Device Encoding Support

BMDStreamingEncodingSupport enumerates the possible types of support for an encoding mode.

bmdStreamingEncodingModeNotSupported
The encoding mode is not supported.

bmdStreamingEncodingModeSupported
The encoding mode is supported.

bmdStreamingEncodingModeSupportedWithChanges
The encoding mode is supported with changes to encoding parameters.

2.7.47 Streaming Device Codecs

BMDStreamingVideoCodec enumerates the possible codecs that are supported by the streaming device.

bmdStreamingVideoCodecH264
The H.264/AVC video compression codec.

2.7.48 Streaming Device H264 Profile

BMDStreamingH264Profile enumerates the possible H.264 video coding profiles that are available on the streaming device. Profiles indicate the complexity of algorithms and coding tools required by a decoder, with Baseline Profile requiring the lowest complexity decoder to decode the encoded video.

bmdStreamingH264ProfileHigh
High Profile

bmdStreamingH264ProfileMain
Main Profile

bmdStreamingH264ProfileBaseline
Baseline Profile

2.7.49 Streaming Device H264 Level

BMDStreamingH264Level enumerates the possible H.264 video coding levels that are available on the streaming device. Levels indicate bitrate and resolution constraints on a video decoder. Higher levels require a decoder capable of decoding higher bitrates and resolutions than lower levels.

bmdStreamingH264Level12
Level 1.2

bmdStreamingH264Level13
Level 1.3

bmdStreamingH264Level2
Level 2

bmdStreamingH264Level21
Level 2.1

bmdStreamingH264Level22
Level 2.2

bmdStreamingH264Level3
Level 3

bmdStreamingH264Level31
Level 3.1

bmdStreamingH264Level32
Level 3.2

bmdStreamingH264Level4
Level 4

bmdStreamingH264Level41
Level 4.1

bmdStreamingH264Level42
Level 4.2

2.7.50 Streaming Device H264 Entropy Coding

BMDStreamingH264EntropyCoding enumerates the possible entropy coding options.

bmdStreamingH264EntropyCodingCAVLC
Context-adaptive variable-length coding.

bmdStreamingH264EntropyCodingCABAC
Context-adaptive binary arithmetic coding.

2.7.51 Streaming Device Audio Codec

BMDStreamingAudioCodec enumerates the possible audio codecs.

bmdStreamingAudioCodecAAC
MPEG Advanced Audio Coding (AAC).

2.7.52 Streaming Device Encoding Mode Properties

BMDStreamingEncodingModePropertyID enumerates the possible properties of the encoding mode.

bmdStreamingEncodingPropertyVideoFrameRate
Video frame rate as a BMDStreamingEncodingFrameRate value

bmdStreamingEncodingPropertyVideoBitRateKbps
Video codec bitrate in kilobits per second

bmdStreamingEncodingPropertyH264Profile
Video codec profile as a BMDStreamingH264Profile value

bmdStreamingEncodingPropertyH264Level
Video codec level as a BMDStreamingH264Level value

bmdStreamingEncodingPropertyH264EntropyCoding
Video codec entropy coding as a BMDStreamingH264EntropyCoding value

bmdStreamingEncodingPropertyH264HasBFrames
Boolean value indicating whether B-Frames will be output by encoding mode

bmdStreamingEncodingPropertyAudioCodec
Audio codec as a BMDStreamingAudioCodec value

bmdStreamingEncodingPropertyAudioSampleRate
Audio sampling rate in Hertz

bmdStreamingEncodingPropertyAudioChannelCount
Number of audio channels

bmdStreamingEncodingPropertyAudioBitRateKbps
Audio codec bitrate in kilobits per second

2.7.53 Audio Formats

BMDAudioFormat enumerates the audio formats supported for encoder capture

bmdAudioFormatPCM : 'lpcm'

Linear signed PCM samples

- Signed PCM samples, see BMDAudioSampleRate for the available sample rates and BMDAudioSampleType for the available sample sizes.

2.7.54 Deck Control Connection

BMDDeckControlConnection enumerates the possible deck control connections.

bmdDeckControlConnectionRS422Remote1
First RS422 deck control connection

bmdDeckControlConnectionRS422Remote2
Second RS422 deck control connection

2.7.55 Video Encoder Frame Coding Mode

BMDVideoEncoderFrameCodingMode enumerates the frame coding mode options.

bmdVideoEncoderFrameCodingModeInter
Video frame data is compressed with reference to neighbouring video frame data.

BmdVideoEncoderFrameCodingModeIntra
Video frame data is compressed relative to the current frame only.

2.7.56 DeckLink Encoder Configuration ID

BMDDeckLinkEncoderConfigurationID enumerates the set of video encoder configuration settings which may be set or queried (see IDeckLinkEncoderConfiguration for details).

Name	Type	Description
bmdDeckLinkEncoderConfigPreferred BitDepth	Int(64)	Video encoder bit depth. Acceptable values are 8, 10, representing 8bit, 10bit respectively.
bmdDeckLinkEncoderConfigFrame CodingMode	Int(64)	Video encoder frame coding mode. See BMDVideoEncoderFrameCodingMode for more information.
bmdDeckLinkEncoderConfigH265TargetBitrate	Int(64)	H.265 target bitrate. Acceptable range is between 2500 (2.5Mbit/s) and 50000000 (50Mbit/s).
bmdDeckLinkEncoderConfigMPEG4 SampleDescription	Bytes	Codec configuration data represented as a full MPEG4 sample description (aka SampleEntry of an 'stsd' atom-box). Useful for MediaFoundation, QuickTime, MKV and more. ***Note:*** The buffer returned by this configuration item is only valid while encoded video input is enabled (i.e.**IDeckLinkEncoderInput::EnableVideoInput** has been called).
bmdDeckLinkEncoderConfigMPEG4Codec SpecificDesc	Bytes	Codec configuration data represented as sample description extensions only (atom stream, each with size and fourCC header). Useful for AVFoundation, VideoToolbox, MKV and more. ***Note:*** The buffer returned by this configuration item is only valid while encoded video input is enabled (i.e.**IDeckLinkEncoderInput::EnableVideoInput** has been called).
bmdDeckLinkEncoderConfigDNxHR CompressionID	Int(64)	DNxHR Compression ID.
bmdDeckLinkEncoderConfigDNxHRLevel	Int(64)	DNxHR Level. BMDDNxHRLevel enumerates the available DNxHR levels.

2.7.57 Device Interface

BMDDeviceInterface enumerates the possible interfaces by which the device is connected.

bmdDeviceInterfacePCI
PCI

bmdDeviceInterfaceUSB
USB

bmdDeviceInterfaceThunderbolt
Thunderbolt

2.7.58 Packet Type

BMDPacketType enumerates the possible IDeckLinkEncoderPacket types.

bmdPacketTypeStreamInterruptedMarker
A packet of this type marks when a video stream was interrupted.

bmdPacketTypeStreamData
Regular stream data.

2.7.59 DeckLink Status ID

BMDDeckLinkStatusID enumerates the set of status information for a DeckLink device which may be queried (see the **IDeckLinkStatus** interface for details).

Name	Type	Description
bmdDeckLinkStatusDetectedVideoInputMode	Int	The detected video input mode (**BMDDisplayMode**), available on devices which support input format detection.
bmdDeckLinkStatusDetectedVideoInputFlags	Int	The detected video input flags (**BMDDeckLinkVideoStatusFlags**), available on devices which support input format detection.
bmdDeckLinkStatusCurrentVideoInputMode	Int	The current video input mode (**BMDDisplayMode**).
bmdDeckLinkStatusCurrentVideoInputPixelFormat	Int	The current video input pixel format (**BMDPixelFormat**).
bmdDeckLinkStatusCurrentVideoInputFlags	Int	The current video input flags (**BMDDeckLinkVideoStatusFlags**)
bmdDeckLinkStatusCurrentVideoOutputMode	Int	The current video output mode (**BMDDisplayMode**).
bmdDeckLinkStatusCurrentVideoOutputFlags	Int	The current video output flags (**BMDDeckLinkVideoStatusFlags**).
bmdDeckLinkStatusPCIExpressLinkWidth	Int	PCIe link width, x1, x4, etc.
bmdDeckLinkStatusPCIExpressLinkSpeed	Int	PCIe link speed, Gen. 1, Gen. 2, etc.
bmdDeckLinkStatusLastVideoOutputPixelFormat	Int	The last video output pixel format (**BMDPixelFormat**).
bmdDeckLinkStatusReferenceSignalMode	Int	The detected reference input mode (**BMDDisplayMode**), available on devices which support reference input format detection.
bmdDeckLinkStatusDuplexMode	Int	The duplex status (**BMDDuplexStatus**) of the DeckLink device / sub-device.
bmdDeckLinkStatusBusy	Int	The current busy state of the device. (See **BMDDeviceBusyState** for more information).
bmdDeckLinkStatusVideoInputSignalLocked	Flag	True if the video input signal is locked.
bmdDeckLinkStatusReferenceSignalLocked	Flag	True if the reference input signal is locked.
bmdDeckLinkStatusReferenceSignalFlags	Int	The detected reference input flags (**BMDDeckLinkVideoStatusFlags**), available on devices which support reference input format detection.
bmdDeckLinkStatusInterchangeablePanelType	Int	The interchangeable panel installed (**BMDPanelType**).
bmdDeckLinkStatusReceivedEDID	Bytes	The received EDID of a connected HDMI sink device.
bmdDeckLinkStatusDeviceTemperature	Int	The on-board temperature (°C).

2.7.60 Duplex Mode

BMDDuplexMode enumerates the duplex mode options for DeckLink sub-devices which support configurable duplex mode.

bmdDuplexModeFull
Configure this sub-device to use two connectors (full-duplex). Note that the paired sub-device will become inactive.

bmdDuplexModeHalf
Configure this sub-device to use a single connector (half-duplex).

2.7.61 Video Status Flags

BMDDeckLinkVideoStatusFlags enumerates status flags associated with a video signal.

bmdDeckLinkVideoStatusPsF
Progressive frames are encoded as PsF.

bmdDeckLinkVideoStatusDualStream3D
The video signal is dual stream 3D video.

2.7.62 Duplex Status

BMDDuplexStatus enumerates the current duplex status.

bmdDuplexStatusFullDuplex
Capable of simultaneous playback and capture.

bmdDuplexStatusHalfDuplex
Capable of playback or capture but not both simultaneously.

bmdDuplexStatusSimplex
Capable of playback only or capture only.

bmdDuplexStatusInactive
Device is inactive.

2.7.63 Frame Metadata ID

BMDDeckLinkFrameMetadataID enumerates the set of video frame metadata which may be queried from the **IDeckLinkVideoFrameMetadataExtensions** interface.

HDR Metadata ID

Name	Type	Description
bmdDeckLinkFrameMetadataHDRElectroOpticalTransferFunc	Int	EOTF in range 0-7 as per CEA 861.3
bmdDeckLinkFrameMetadataHDRDisplayPrimariesRedX	Float	Red display primaries in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRDisplayPrimariesRedY	Float	Red display primaries in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRDisplayPrimariesGreenX	Float	Green display primaries in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRDisplayPrimariesGreenY	Float	Green display primaries in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRDisplayPrimariesBlueX	Float	Blue display primaries in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRDisplayPrimariesBlueY	Float	Blue display primaries in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRWhitePointX	Float	White point in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRWhitePointY	Float	White point in range 0.0 - 1.0
bmdDeckLinkFrameMetadataHDRMaxDisplay MasteringLuminance	Float	Max display mastering luminance in range 1 cd/m2 - 65535 cd/m2
bmdDeckLinkFrameMetadataHDRMinDisplay MasteringLuminance	Float	Min display mastering luminance in range 0.0001 cd/m2 - 6.5535 cd/m2
bmdDeckLinkFrameMetadataHDRMaximum ContentLightLevel	Float	Maximum Content Light Level in range 1 cd/m2 - 65535 cd/m2
bmdDeckLinkFrameMetadataHDRMaximumFrame AverageLightLevel	Float	Maximum Frame Average Light Level in range 1 cd/m2 - 65535 cd/m2
bmdDeckLinkFrameMetadataColorspace	Int	Colorspace of video frame (see **BMDColorspace**)

Cintel Metadata ID

Name	Type	Description
bmdDeckLinkFrameMetadataCintelFilmType	Int	Current film type
bmdDeckLinkFrameMetadataCintelFilmGauge	Int	Current film gauge
bmdDeckLinkFrameMetadataCintelOffsetDetectedHorizontal	Int	Horizontal offset (pixels) detected in image
bmdDeckLinkFrameMetadataCintelOffsetDetectedVertical	Int	Vertical offset (pixels) detected in image
bmdDeckLinkFrameMetadataCintelKeykodeLow	Int	Raw keykode value - low 64 bits
bmdDeckLinkFrameMetadataCintelKeykodeHigh	Int	Raw keykode value - high 64 bits
bmdDeckLinkFrameMetadataCintelTile1Size	Int	Size in bytes of compressed raw tile 1
bmdDeckLinkFrameMetadataCintelTile2Size	Int	Size in bytes of compressed raw tile 2
bmdDeckLinkFrameMetadataCintelTile3Size	Int	Size in bytes of compressed raw tile 3
bmdDeckLinkFrameMetadataCintelTile4Size	Int	Size in bytes of compressed raw tile 4
bmdDeckLinkFrameMetadataCintelImageWidth	Int	Width in pixels of image
bmdDeckLinkFrameMetadataCintelImageHeight	Int	Height in pixels of image
bmdDeckLinkFrameMetadataCintelLinearMaskingRedInRed	Int	Red in red linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingGreenInRed	Int	Green in red linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingBlueInRed	Int	Blue in red linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingRedInGreen	Int	Red in green linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingGreenInGreen	Int	Green in green linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingBlueInGreen	Int	Blue in green linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingRedInBlue	Int	Red in blue linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingGreenInBlue	Int	Green in blue linear masking parameter
bmdDeckLinkFrameMetadataCintelLinearMaskingBlueInBlue	Int	Blue in blue linear masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingRedInRed	Int	Red in red log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingGreenInRed	Int	Green in red log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingBlueInRed	Int	Blue in red log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingRedInGreen	Int	Red in green log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingGreenInGreen	Int	Green in green log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingBlueInGreen	Int	Blue in green log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingRedInBlue	Int	Red in blue log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingGreenInBlue	Int	Green in blue log masking parameter
bmdDeckLinkFrameMetadataCintelLogMaskingBlueInBlue	Int	Blue in blue log masking parameter

Name	Type	Description
bmdDeckLinkFrameMetadataCintelFilmFrameRate	Int	Film frame rate
bmdDeckLinkFrameMetadataCintelOffsetToApplyHorizontal	Float	Horizontal offset (pixels) to be applied to image
bmdDeckLinkFrameMetadataCintelOffsetToApplyVertical	Float	Vertical offset (pixels) to be applied to image
bmdDeckLinkFrameMetadataCintelGainRed	Float	Red gain parameter to apply after log
bmdDeckLinkFrameMetadataCintelGainGreen	Float	Green gain parameter to apply after log
bmdDeckLinkFrameMetadataCintelGainBlue	Float	Blue gain parameter to apply after log
bmdDeckLinkFrameMetadataCintelLiftRed	Float	Red lift parameter to apply after log and gain
bmdDeckLinkFrameMetadataCintelLiftGreen	Float	Green lift parameter to apply after log and gain
bmdDeckLinkFrameMetadataCintelLiftBlue	Float	Blue lift parameter to apply after log and gain

2.7.64 DNxHR Levels

BMDDNxHRLevel enumerates the available DNxHR levels.

bmdDNxHRLevelSQ
DNxHR Standard Quality

bmdDNxHRLevelLB
DNxHR Low Bandwidth

bmdDNxHRLevelHQ
DNxHR High Quality (8 bit)

bmdDNxHRLevelHQX
DNxHR High Quality (12 bit)

bmdDNxHRLevel444
DNxHR 4:4:4

2.7.65 Panel Type

BMDPanelType enumerates the type of interchangeable panel installed

bmdPanelNotDetected
No panel detected

bmdPanelTeranexMiniSmartPanel
Teranex Mini Smart Panel detected

2.7.66 Ancillary Packet Format

BMDAncillaryPacketFormat enumerates the possible data formats of the ancillary packet.

bmdAncillaryPacketFormatUInt8
8-bit unsigned integer

bmdAncillaryPacketFormatUInt16
16-bit unsigned integer

bmdAncillaryPacketFormatYCbCr10
Native v210 pixel format (see bmdFormat10BitYUV for packing structure).

2.7.67 Colorspace

BMDColorspace enumerates the colorspace for a video frame.

bmdColorspaceRec601
Rec. 601 colorspace

bmdColorspaceRec709
Rec. 709 colorspace

bmdColorspaceRec2020
Rec. 2020 colorspace