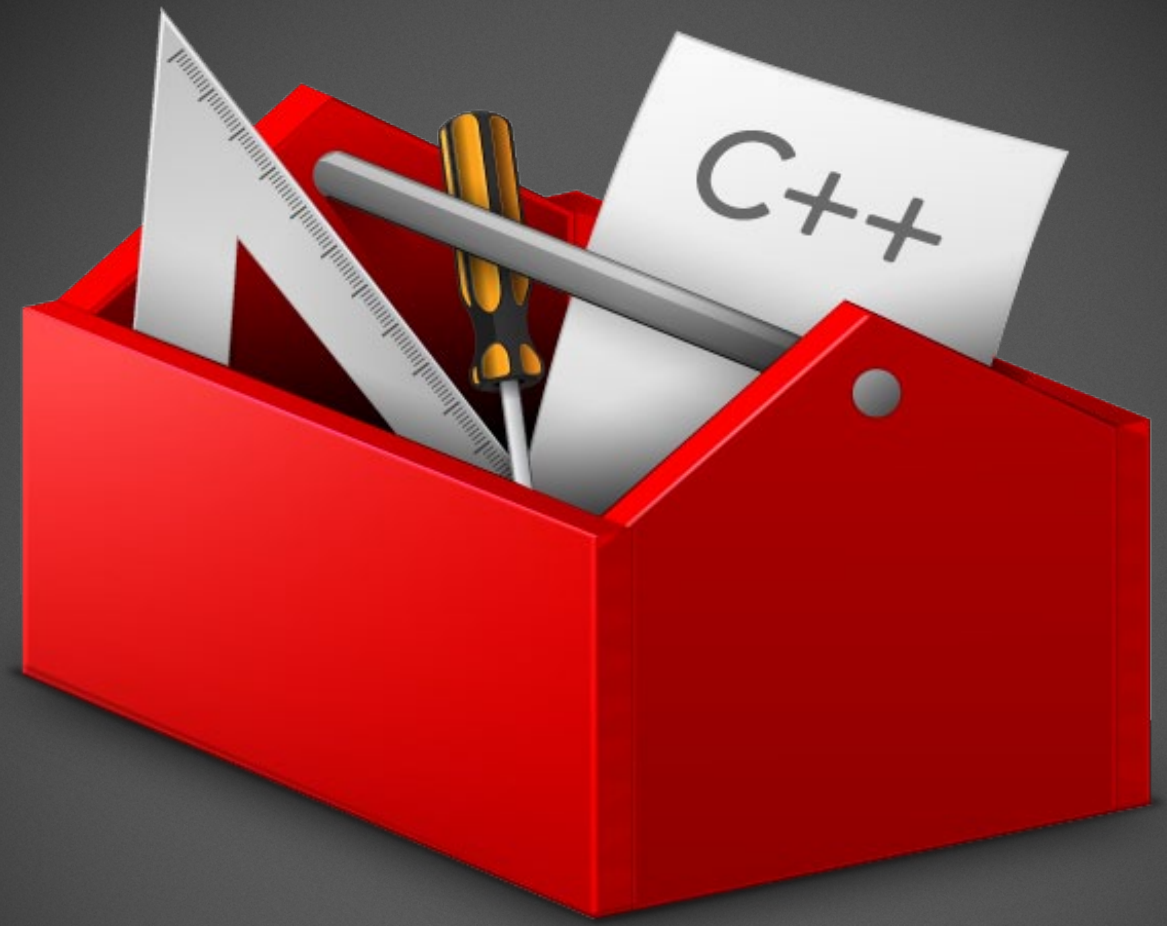
SDK – DeckLink
Software Developers Kit

Blackmagicdesign



Mac OS X™

Windows™

Linux™

July 2011

TABLE OF CONTENTS

1 Introduction 10

1.1 Welcome 10
1.2 Overview 10
1.3 API Design 11
1.3.1 Overview 11
1.3.2 Object Model 11
1.3.3 Object Interfaces 11
1.3.4 Reference Counting 12
1.3.5 Interface Stability 12
1.3.5.1 New Interfaces 12
1.3.5.2 Updated Interfaces 13
1.3.5.3 Deprecated Interfaces 13
1.3.5.4 Removed interfaces 13
1.4 Interface Reference 14
1.4.1 IUnknown Interface 14
1.4.1.1 IUnknown::QueryInterface method 15
1.4.1.2 IUnknown::AddRef method 16
1.4.1.3 IUnknown::Release method 16

2 DeckLink API 17

2.1 Using the DeckLink API in a project 17
2.2 Accessing DeckLink devices 17
2.2.1 Windows 18
2.2.2 Mac OS X and Linux 18
2.3 High level interface 19
2.3.1 Capture 19
2.3.2 Playback 20
2.3.3 3D Functionality 21
2.3.3.1 3D Capture 21
2.3.3.2 3D Playback 23
2.4 Interface Reference 25
2.4.1 IDeckLinkIterator Interface 25
2.4.1.1 IDeckLinkIterator::Next method 26
2.4.2 IDeckLink Interface 27

TABLE OF CONTENTS

2.4.2.1	IDeckLink::GetModelName method	29
2.4.2.2	IDeckLink::GetDisplayName method	30
2.4.3	IDeckLinkOutput Interface	31
2.4.3.1	IDeckLinkOutput::DoesSupportVideoMode method	34
2.4.3.2	IDeckLinkOutput::IsScheduledPlaybackRunning method	35
2.4.3.3	IDeckLinkOutput::GetDisplayModeIterator method	36
2.4.3.4	IDeckLinkOutput::SetScreenPreviewCallback method	37
2.4.3.5	IDeckLinkOutput::EnableVideoOutput method	38
2.4.3.6	IDeckLinkOutput::DisableVideoOutput method	38
2.4.3.7	IDeckLinkOutput::SetVideoOutputFrameMemoryAllocator method	39
2.4.3.8	IDeckLinkOutput::CreateVideoFrame method	40
2.4.3.9	IDeckLinkOutput::CreateAncillaryData method	41
2.4.3.10	IDeckLinkOutput::DisplayVideoFrameSync method	42
2.4.3.11	IDeckLinkOutput::ScheduleVideoFrame method	43
2.4.3.12	IDeckLinkOutput::SetScheduledFrameCompletionCallback method	44
2.4.3.13	IDeckLinkOutput::GetBufferedVideoFrameCount method	45
2.4.3.14	IDeckLinkOutput::EnableAudioOutput method	46
2.4.3.15	IDeckLinkOutput::DisableAudioOutput method	47
2.4.3.16	IDeckLinkOutput::WriteAudioSamplesSync method [currently not supported]	48
2.4.3.17	IDeckLinkOutput::BeginAudioPreroll method	49
2.4.3.18	IDeckLinkOutput::EndAudioPreroll method	50
2.4.3.19	IDeckLinkOutput::ScheduleAudioSamples method	51
2.4.3.20	IDeckLinkOutput::GetBufferedAudioSampleFrameCount	52
2.4.3.21	IDeckLinkOutput::FlushBufferedAudioSamples method	53
2.4.3.22	IDeckLinkOutput::SetAudioCallback method	54
2.4.3.23	IDeckLinkOutput::StartScheduledPlayback method	55
2.4.3.24	IDeckLinkOutput::StopScheduledPlayback method	56
2.4.3.25	IDeckLinkOutput::GetScheduledStreamTime method	57
2.4.3.26	IDeckLinkOutput::GetReferenceStatus method	58
2.4.3.27	IDeckLinkOutput::GetHardwareReferenceClock method	59
2.4.4	IDeckLinkInput Interface	60
2.4.4.1	IDeckLinkInput::DoesSupportVideoMode method	62
2.4.4.2	IDeckLinkInput::GetDisplayModeIterator method	63
2.4.4.3	IDeckLinkInput::SetScreenPreviewCallback method	64
2.4.4.4	IDeckLinkInput::EnableVideoInput method	65

TABLE OF CONTENTS

2.4.4.5	IDeckLinkInput::GetAvailableVideoFrameCount method	66
2.4.4.6	IDeckLinkInput::DisableVideoInput method	67
2.4.4.7	IDeckLinkInput::EnableAudioInput method	68
2.4.4.8	IDeckLinkInput::DisableAudioInput method	69
2.4.4.9	IDeckLinkInput::GetAvailableAudioSampleFrameCount method	70
2.4.4.10	IDeckLinkInput::StartStreams method	71
2.4.4.11	IDeckLinkInput::StopStreams method	71
2.4.4.12	IDeckLinkInput::FlushStreams method	72
2.4.4.13	IDeckLinkInput::PauseStreams method	73
2.4.4.14	IDeckLinkInput::SetCallback method	74
2.4.4.15	IDeckLinkInput::GetHardwareReferenceClock method	75
2.4.5	IDeckLinkVideoFrame Interface	76
2.4.5.1	IDeckLinkVideoFrame::GetWidth method	78
2.4.5.2	IDeckLinkVideoFrame::GetHeight method	78
2.4.5.3	IDeckLinkVideoFrame::GetRowBytes method	79
2.4.5.4	IDeckLinkVideoFrame::GetPixelFormat method	79
2.4.5.5	IDeckLinkVideoFrame::GetFlags method	80
2.4.5.6	IDeckLinkVideoFrame::GetBytes method	80
2.4.5.7	IDeckLinkVideoFrame::GetTimecode method	81
2.4.5.8	IDeckLinkVideoFrame::GetAncillaryData method	82
2.4.6	IDeckLinkVideoOutputCallback Interface	83
2.4.6.1	IDeckLinkVideoOutputCallback::ScheduledFrameCompleted method	84
2.4.6.2	IDeckLinkVideoOutputCallback::ScheduledPlaybackHasStopped method	85
2.4.7	IDeckLinkMutableVideoFrame Interface	86
2.4.7.1	IDeckLinkMutableVideoFrame::SetFlags method	87
2.4.7.2	IDeckLinkMutableVideoFrame::SetTimecode method	88
2.4.7.3	IDeckLinkMutableVideoFrame::SetTimecodeFromComponents method	89
2.4.7.4	IDeckLinkMutableVideoFrame::SetAncillaryData method	90
2.4.7.5	IDeckLinkMutableVideoFrame::SetTimecodeUserBits method	91
2.4.8	IDeckLinkVideoFrame3DExtensions Interface	92
2.4.8.1	IDeckLinkVideoFrame3DExtensions::Get3DPackingFormat method	93
2.4.8.2	IDeckLinkVideoFrame3DExtensions::GetFrameForRightEye method	94
2.4.9	IDeckLinkAudioOutputCallback Interface	95
2.4.9.1	IDeckLinkAudioOutputCallback::RenderAudioSamples method	96
2.4.10	IDeckLinkInputCallback Interface	97

TABLE OF CONTENTS

2.4.10.1	IDeckLinkInputCallback::VideoInputFrameArrived method	98
2.4.10.2	IDeckLinkInputCallback::VideoInputFormatChanged method	100
2.4.11	IDeckLinkVideoInputFrame Interface	101
2.4.11.1	IDeckLinkVideoInputFrame::GetStreamTime method	102
2.4.11.2	IDeckLinkVideoInputFrame::GetHardwareReferenceTimestamp method	103
2.4.12	IDeckLinkAudioInputPacket Interface	104
2.4.12.1	IDeckLinkAudioInputPacket:: GetSampleFrameCount method	105
2.4.12.2	IDeckLinkAudioInputPacket::GetBytes method	105
2.4.12.3	IDeckLinkAudioInputPacket::GetPacketTime method	106
2.4.13	IDeckLinkDisplayModeIterator Interface	107
2.4.13.1	IDeckLinkDisplayModeIterator::Next method	108
2.4.14	IDeckLinkDisplayMode Interface	109
2.4.14.1	IDeckLinkDisplayMode::GetWidth method	110
2.4.14.2	IDeckLinkDisplayMode::GetHeight method	110
2.4.14.3	IDeckLinkDisplayMode::GetName method	111
2.4.14.4	IDeckLinkDisplayMode::GetDisplayMode method	111
2.4.15	IDeckLinkConfiguration Interface	112
2.4.15.1	IDeckLinkConfiguration::SetFlag method	114
2.4.15.2	IDeckLinkConfiguration::GetFlag method	115
2.4.15.3	IDeckLinkConfiguration::SetInt method	116
2.4.15.4	IDeckLinkConfiguration::GetInt method	117
2.4.15.5	IDeckLinkConfiguration::SetFloat method	118
2.4.15.6	IDeckLinkConfiguration::GetFloat method	119
2.4.15.7	IDeckLinkConfiguration::SetString method	120
2.4.15.8	IDeckLinkConfiguration::GetString method	121
2.4.15.9	IDeckLinkConfiguration::WriteConfigurationToPreferences method	122
2.4.16	IDeckLinkAPIInformation Interface	123
2.4.16.1	IDeckLinkAPIInformation::GetFlag method	124
2.4.16.2	IDeckLinkAPIInformation::GetInt method	125
2.4.16.3	IDeckLinkAPIInformation::GetFloat method	126
2.4.16.4	IDeckLinkAPIInformation::GetString method	127
2.4.17	IDeckLinkAttributes Interface	128
2.4.17.1	IDeckLinkAttributes::GetFlag method	129
2.4.17.2	IDeckLinkAttributes::GetInt method	130
2.4.17.3	IDeckLinkAttributes::GetFloat method	131

TABLE OF CONTENTS

2.4.17.4	IDeckLinkAttributes::GetString method	132
2.4.18	IDeckLinkMemoryAllocator Interface	133
2.4.18.1	IDeckLinkMemoryAllocator::AllocateBuffer method	134
2.4.18.2	IDeckLinkMemoryAllocator::ReleaseBuffer method	135
2.4.18.3	IDeckLinkMemoryAllocator::Commit method	136
2.4.18.4	IDeckLinkMemoryAllocator::Decommit method	137
2.4.19	IDeckLinkKeyer Interface	138
2.4.19.1	IDeckLinkKeyer::Enable method	139
2.4.19.2	IDeckLinkKeyer::SetLevel method	140
2.4.19.3	IDeckLinkKeyer::RampUp method	141
2.4.19.4	IDeckLinkKeyer::RampDown method	142
2.4.19.5	IDeckLinkKeyer::Disable method	143
2.4.20	IDeckLinkVideoFrameAncillary Interface	144
2.4.20.1	IDeckLinkVideoFrameAncillary GetPixelFormat method	145
2.4.20.2	IDeckLinkVideoFrameAncillary GetDisplayMode method	145
2.4.20.3	IDeckLinkVideoFrameAncillary GetBufferForVerticalBlankingLine method	146
2.4.21	IDeckLinkTimecode Interface	147
2.4.21.1	IDeckLinkTimecode::GetBCD method	148
2.4.21.2	IDeckLinkTimecode::GetComponents method	149
2.4.21.3	IDeckLinkTimecode::GetString method	150
2.4.21.4	IDeckLinkTimecode::GetFlags method	150
2.4.21.5	IDeckLinkTimecode::GetTimecodeUserBits method	151
2.4.22	IDeckLinkScreenPreviewCallback Interface	152
2.4.22.1	IDeckLinkScreenPreviewCallback::DrawFrame method	153
2.4.23	IDeckLinkGLScreenPreviewHelper Interface	154
2.4.23.1	IDeckLinkGLScreenPreviewHelper::InitializeGL method	156
2.4.23.2	IDeckLinkGLScreenPreviewHelper::PaintGL method	156
2.4.23.3	IDeckLinkGLScreenPreviewHelper::SetFrame method	157
2.4.23.4	IDeckLinkGLScreenPreviewHelper::Set3DPreviewFormat	158
2.4.24	IDeckLinkCocoaScreenPreviewCallback Interface	159
2.4.25	IDeckLinkVideoConversion Interface	160
2.4.25.1	IDeckLinkVideoConversion::ConvertFrame method	160
2.4.26	IDeckLinkDeckControl Interface	162
2.4.26.1	IDeckLinkDeckControl::Open method	165
2.4.26.2	IDeckLinkDeckControl::Close method	166

TABLE OF CONTENTS

2.4.26.3	IDeckLinkDeckControl::GetCurrentState method	167
2.4.26.4	IDeckLinkDeckControl::SetStandby method	168
2.4.26.5	IDeckLinkDeckControl::SendCommand method	169
2.4.26.6	IDeckLinkDeckControl::Play method	170
2.4.26.7	IDeckLinkDeckControl::Stop method	171
2.4.26.8	IDeckLinkDeckControl::TogglePlayStop method	172
2.4.26.9	IDeckLinkDeckControl::Eject method	173
2.4.26.10	IDeckLinkDeckControl::GoToTimecode method	174
2.4.26.11	IDeckLinkDeckControl::FastForward method	175
2.4.26.12	IDeckLinkDeckControl::Rewind method	176
2.4.26.13	IDeckLinkDeckControl::StepForward method	177
2.4.26.14	IDeckLinkDeckControl::StepBack method	178
2.4.26.15	IDeckLinkDeckControl::Jog method	179
2.4.26.16	IDeckLinkDeckControl::Shuttle method	180
2.4.26.17	IDeckLinkDeckControl::GetTimecodeString method	181
2.4.26.18	IDeckLinkDeckControl::GetTimecode method	182
2.4.26.19	IDeckLinkDeckControl::GetTimecodeBCD method	183
2.4.26.20	IDeckLinkDeckControl::SetPreroll method	184
2.4.26.21	IDeckLinkDeckControl::GetPreroll method	185
2.4.26.22	IDeckLinkDeckControl::SetCaptureOffset method	186
2.4.26.23	IDeckLinkDeckControl::GetCaptureOffset method	186
2.4.26.24	IDeckLinkDeckControl::SetExportOffset method	188
2.4.26.25	IDeckLinkDeckControl::GetExportOffset method	189
2.4.26.26	IDeckLinkDeckControl::GetManualExportOffset method	190
2.4.26.27	IDeckLinkDeckControl::StartExport method	191
2.4.26.28	IDeckLinkDeckControl::StartCapture method	193
2.4.26.29	IDeckLinkDeckControl::GetDeviceID method	195
2.4.26.30	IDeckLinkDeckControl::Abort method	196
2.4.26.31	IDeckLinkDeckControl::CrashRecordStart method	197
2.4.26.32	IDeckLinkDeckControl::CrashRecordStop method	198
2.4.26.33	IDeckLinkDeckControl::SetCallback method	199
2.4.27	IDeckLinkDeckControlStatusCallback Interface	200
2.4.27.1	IDeckLinkDeckControlStatusCallback::TimecodeUpdate method	201
2.4.27.2	IDeckLinkDeckControlStatusCallback::VTRControlStateChanged method	202
2.4.27.3	IDeckLinkDeckControlStatusCallback::DeckControlEventReceived method	203

TABLE OF CONTENTS

2.4.27.4	IDeckLinkDeckControlStatusCallback::DeckControlStatusChanged method	204
2.5	Common Data Types	205
2.5.1	Basic Types	205
2.5.2	Time Representation	205
2.5.3	Display Modes	206
2.5.4	Pixel Formats	208
2.5.5	Field Dominance	212
2.5.6	Frame Flags	213
2.5.7	Video Input Flags	213
2.5.8	Video Output Flags	213
2.5.9	Output Frame Completion Results Flags	214
2.5.10	Frame preview format	214
2.5.11	Video Connection Modes	215
2.5.12	Audio Sample Rates	215
2.5.13	Audio Sample Types	215
2.5.14	DeckLink Information ID	216
2.5.15	DeckLink Attribute ID	217
2.5.16	DeckLink Configuration ID	219
2.5.17	Audio Output Stream Type	224
2.5.18	Analog Video Flags	224
2.5.19	Audio Connection Modes	224
2.5.20	Audio Output Selection switch	225
2.5.21	Output Conversion Modes	225
2.5.22	Input Conversion Modes	226
2.5.23	Video Input Format Changed Events	226
2.5.24	Detected Video Input Format Flags	226
2.5.25	Display Mode Characteristics	227
2.5.26	Video 3D packing format	227
2.5.27	Display Mode Support	227
2.5.28	BMDTimecodeFormat	228
2.5.29	BMDTimecodeFlags	228
2.5.30	BMDTimecodeBCD	229
2.5.31	Deck Control Mode	230
2.5.32	Deck Control Event	230
2.5.33	Deck Control VTR Control States	231

TABLE OF CONTENTS

2.5.34	Deck Control Status Flags	231
2.5.35	Deck Control Export Mode Ops Flags	232
2.5.36	Deck Control error	233
2.5.37	Genlock reference status	234
2.5.38	Idle Video Output Operation	234
2.5.39	Device Busy State	234
2.6	Formulas	235
2.6.1	Color space Conversion Formulas	235

SECTION 1

INTRODUCTION

1.1 Welcome

Thanks for downloading the Blackmagic Design DeckLink Software Developers Kit.

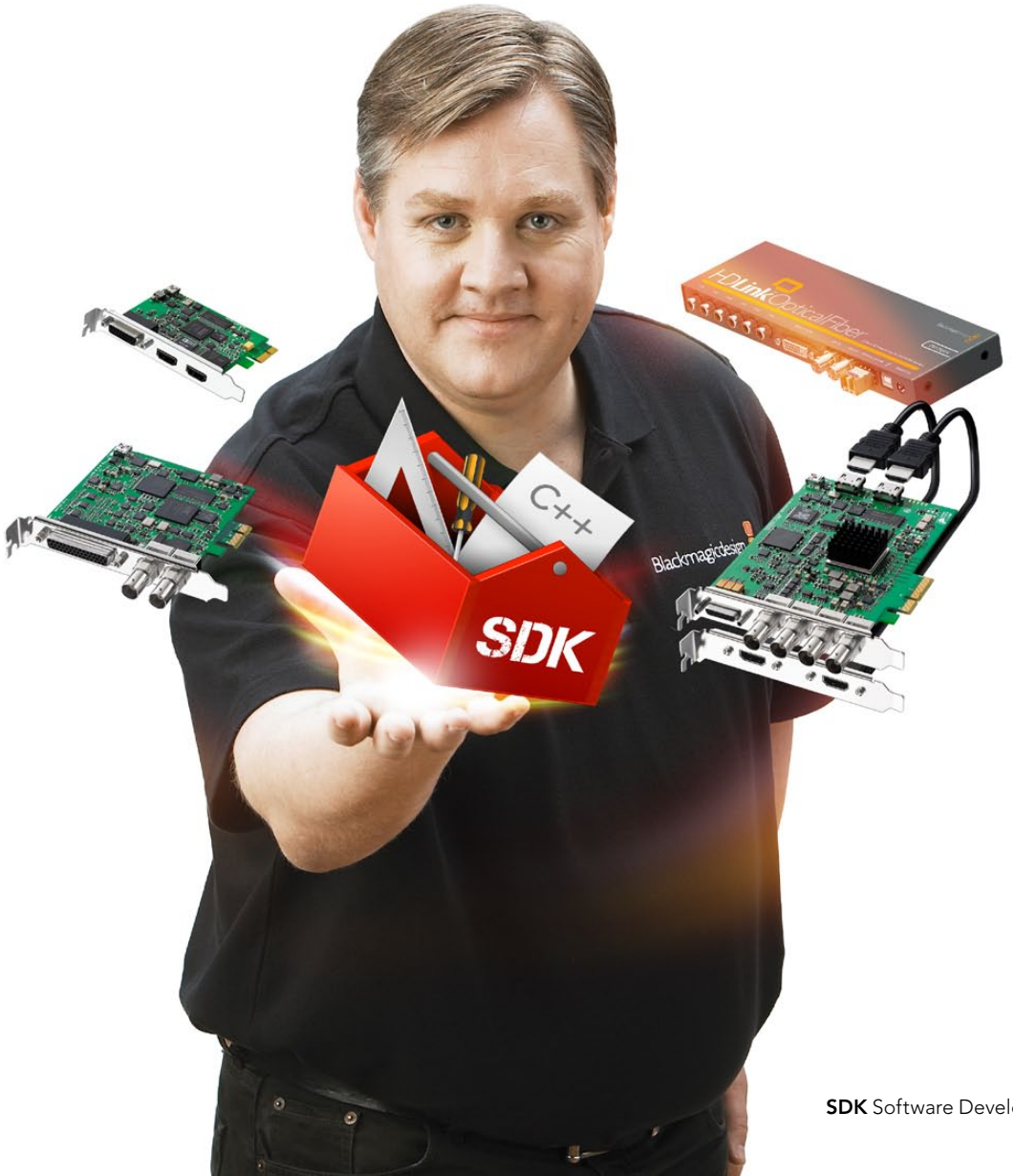
1.2 Overview

The DeckLink SDK provides a stable, cross- platform interface to a Blackmagic Design DeckLink product line. 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 XP, Vista, Mac OS X and Linux platforms.

You can download the SDK from **www.blackmagic-design.com/support**



SECTION 1

API DESIGN

1.3.1 Overview

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.

1.3.2 Object Model

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 without imposing much overhead or baggage.

1.3.3 Object Interfaces

The DeckLink API provides programmatic access to all current Blackmagic Design DeckLink, Multibridge and Intensity products. "DeckLink" is used as a generic term to refer to all supported products and models except where noted.

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.

1.3.4 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.

1.3.5 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.

1.3.5.1 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.

1.3.5.2 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.

1.3.5.3 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.

1.3.5.4 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
`outputInterface`	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();
```

Parameters

none.

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();
```

Parameters

none.

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
Mac OS X	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 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:

SECTION 2 DeckLink API

2.2.1 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.

2.2.2 Mac OS X 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.

2.3 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.

2.3.1 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 **IDeckLinkInputCallbackVideoInputFrameArrived** callback will receive NULL audio packets.

2.3.2 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.

2.3.3 3D Functionality

3D (dual-stream) capture and playback is supported by certain DeckLink devices such as the DeckLink HD Extreme 3D. 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.

2.3.3.1 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**

2.3.3.2 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**. **QueryInterface**(**IID_IDeckLinkVideoFrame3DExtensions**) method is called
- 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::BeginAudio Preroll** and **IDeckLinkOutput::EndAudioPreroll** may be omitted.

2.4 Interface Reference

2.4.1 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 a an **IDeckLink** object interface corresponding to an individual DeckLink device.

2.4.1.1 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

2.4.2 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**
IDeckLink Configuration	IID_IDeckLinkConfiguration	An **IDeckLinkConfiguration** object interface may be obtained from **IDeckLink** using **QueryInterface**
IDeckLink Attributes	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**.

Interface	Interface ID	Description
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

2.4.2.1 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

2.4.2.2 IDeckLink::GetDisplayName method

The **GetDisplayName** method returns a string suitable for display in a user interface. 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

2.4.3 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.

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

Public Member Functions	
Method	Description
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

2.4.3.1 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.

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

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

2.4.3.2 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

2.4.3.3 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

2.4.3.4 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

2.4.3.5 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

2.4.3.6 IDeckLinkOutput::DisableVideoOutput method

The **DisableVideoOutput** method disables video output.

Syntax

```
HRESULT        DisableVideoOutput (BMDDisplayMode displayMode);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.3.7 IDeckLinkOutput::SetVideoOutputFrameMemoryAllocator method

The **SetVideoOutputFrameMemoryAllocator** method sets a custom memory allocator for frame allocations. 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

2.4.3.8 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

2.4.3.9 IDeckLinkOutput::CreateAncillaryData method

The **CreateAncillaryData** method creates an ancillary buffer that can be attached to an **IDeckLinkMutable** video frame.

Syntax

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

Parameters

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

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.3.10 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.

2.4.3.11 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

2.4.3.12 IDeckLinkOutput::SetScheduledFrameCompletionCallback method

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

Syntax

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

Parameters

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

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.3.13 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

2.4.3.14 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

2.4.3.15 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

2.4.3.16 IDeckLinkOutput::WriteAudioSamplesSync method [currently not supported]

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

2.4.3.17 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

2.4.3.18 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

2.4.3.19 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.
`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.

2.4.3.20 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

2.4.3.21 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

2.4.3.22 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

2.4.3.23 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

2.4.3.24 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

2.4.3.25 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

2.4.3.26 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

2.4.3.27 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

2.4.4 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**.

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.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink`	`IID_IDeckLink`	An **IDeckLinkInput** object interface may be obtained from **IDeckLink** using **QueryInterface**
`IDeckLinkDisplayModeIterator`	`IID_IDeckLinkDisplayModeIterator`	**IDeckLinkInput::GetDisplayModeIterator** returns an **IDeckLinkDisplayModeIterator** object interface
`IDeckLinkInput-Callback`	`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
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

2.4.4.1 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

2.4.4.2 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

2.4.4.3 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

2.4.4.4 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

2.4.4.5 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

2.4.4.6 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

2.4.4.7 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

2.4.4.8 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

2.4.4.9 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

2.4.4.10 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.

2.4.4.11 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.

2.4.4.12 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

2.4.4.13 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

2.4.4.14 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

2.4.4.15 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

2.4.5 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
`IDeckLinkMutable`	`IID_IDeckLinkMutable`	**IDeckLinkMutable** subclasses **IDeckLinkVideoFrame**
`IDeckLink VideoInputFrame`	`IID_IDeckLink VideoInputFrame`	**IDeckLinkVideoInputFrame** subclasses **IDeckLinkVideoFrame**

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

Public Member Functions	
Method	Description
GetFlags	Get frame flags
GetBytes	Get pointer to frame data
GetTimecode	Gets timecode information
GetAncillaryData	Gets ancillary data

2.4.5.1 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

2.4.5.2 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

2.4.5.3 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

2.4.5.4 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**)

2.4.5.5 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)**

2.4.5.6 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

2.4.5.7 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.

2.4.5.8 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.

2.4.6 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:: SetScheduledFrame CompletionCallback**

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

2.4.6.1 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

2.4.6.2 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

2.4.7 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
`IDeckLink VideoFrame`	`IID_IDeckLink VideoFrame`	**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

2.4.7.1 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

2.4.7.2 IDeckLinkMutableVideoFrame::SetTimecode method

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

Note: To set VITC an ancillary buffer must first be attached.

Syntax

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

Parameters

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

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.7.3 IDeckLinkMutableVideoFrame::SetTimecodeFromComponents method

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

Note: To set VITC an ancillary buffer must first be attached.

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

2.4.7.4 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

2.4.7.5 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**

2.4.8 IDeckLinkVideoFrame3DExtensions Interface

The **IDeckLinkVideoFrame3DExtenstions** 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 a the DeckLink HD Extreme 3D. 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.

2.4.8.1 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.

2.4.8.2 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

2.4.9 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

2.4.9.1 IDeckLinkAudioOutputCallback::RenderAudioSamples method

The **RenderAudioSamples** method is called at a rate of 50Hz during preroll and playback. 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 method should 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

2.4.10 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**
`IDeckLinkVideo InputFrame`	`IID_DeckLinkVideoInput-Frame`	**An IDeckLinkVideoInputFrame** object interface is passed to **IDeckLinkInput Callback::VideoInputFrameArrived**
`IDeckLinkAudio-InputPacket`	`IID_DeckLinkAudioInput-Packet`	An **IDeckLinkAudioInputPacket** object interface is passed to **IDeckLinkInput Callback::VideoInputFrameArrived**

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

2.4.10.1 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.

Name	Direction	Description
`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

2.4.10.2 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

2.4.11 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

2.4.11.1 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

2.4.11.2 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

2.4.12 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
`IDeckLink InputCallback`	`IID_IDeckLink InputCallback`	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

2.4.12.1 IDeckLinkAudioInputPacket::GetSampleFrameCount method

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

Syntax

```
Long        GetSampleCount ();
```

Return Values

Value	Description
Count	Audio packet size in sample frames

2.4.12.2 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

2.4.12.3 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

2.4.13 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
`IDeckLinkInputCallback`	`IID_IDeckLinkInputCallback`	**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

2.4.13.1 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

2.4.14 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
`IDeckLink DisplayMode Iterator`	`IID_IDeckLink DisplayModeIterator`	**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).

2.4.14.1 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

2.4.14.2 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

2.4.14.3 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

2.4.14.4 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

2.4.15 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**.

Public Member Functions	
Method	Description
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.

2.4.15.1 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.

2.4.15.2 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.

2.4.15.3 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.

2.4.15.4 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.

2.4.15.5 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.

2.4.15.6 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.

2.4.15.7 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.

2.4.15.8 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.

2.4.15.9 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.

2.4.16 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**

2.4.16.1 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.

2.4.16.2 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.

2.4.16.3 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.

2.4.16.4 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

2.4.17 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**

2.4.17.1 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.

2.4.17.2 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.

2.4.17.3 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.

2.4.17.4 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.

2.4.18 IDeckLinkMemoryAllocator Interface

The **IDeckLinkMemoryAllocator** interface may be registered as a callback with the **IDeckLinkOutput** interface. **IDeckLinkOutput** will call back to the interface for frame memory management. 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**

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).

2.4.18.1 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.

2.4.18.2 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

2.4.18.3 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.

2.4.18.4 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

2.4.19 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

2.4.19.1 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.

The following table displays hardware which supports the keyer functionality.

Device	Internal	External	HD
Intensity Pro	no	no	no
Intensity Shuttle	no	no	no
DeckLink SDI	yes	no	no
DeckLink Optical Fiber	yes	no	no
DeckLink Duo	yes	no	no
DeckLink Studio	yes	yes	no
UltraStudio Pro	yes	yes	no
DeckLink HD Extreme	yes	yes	yes
Multibridge Pro	yes	yes	yes
Multibridge Eclipse	yes	yes	yes

continued over page...

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

2.4.19.2 DeckLinkKeyer::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

2.4.19.3 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

2.4.19.4 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

2.4.19.5 IDeckLinkKeyer::Disable method

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

Syntax

```
HRESULT        Disable();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.20 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.

Related Interfaces

Interface	Interface ID	Description
`IDeckLink Timecode`	`IID_IDeckLinkTimeCode`	**IDeckLinkVideoFrameAncillary::Get TimeCode** and **IDeckLinkVideoFrame Ancillary::SetTimeCode**, get and set timecodes in ancillary data from an **IDeckLinkTimecode** object interface.

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.

2.4.20.1 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**)

2.4.20.2 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.

2.4.20.3 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

2.4.21 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.

2.4.21.1 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)

2.4.21.2 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

2.4.21.3 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

2.4.21.4 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)

2.4.21.5 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

2.4.22 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 callbackwith 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

2.4.22.1 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

2.4.23 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 **IDeckLinkGLScreenPreview Helper::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

2.4.23.1 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

2.4.23.2 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

2.4.23.3 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 no 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

2.4.23.4 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

2.4.24 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 **IDeckLinkCocoaScreenPreview Callback** object interface may be registered with **IDeckLinkInput::Set ScreenPreviewCallback**
`IDeckLinkOutput`	`IID_IDeckLinkOutput`	An **IDeckLinkCocoaScreenPreview Callback** object interface may be registered with **IDeckLinkOutput::Set ScreenPreviewCallback**

2.4.25 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.

2.4.25.1 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 destination frame with the desired properties should be created using **IDeckLinkOutput::CreateVideoFrame**. The created frame can then be passed into this method.

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

2.4.26 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
`IDeckLink DeckControl`	`IID_IDeckLinkDeck Control`	An **IDecklinkDeckControl** object interface may be obtained from **IDeckLink** using **QueryInterface**.
`IDeckLink DeckControl StatusCallback`	`IID_ IDeckLinkDeck ControlStatusCallback`	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.

Public Member Functions	
Method	Description
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.

Public Member Functions	
Method	Description
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.
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.

2.4.26.1 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.

Syntax

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

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.

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

2.4.26.2 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

2.4.26.3 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.

2.4.26.4 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

2.4.26.5 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

2.4.26.6 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.

2.4.26.7 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.

2.4.26.8 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);
```

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

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

2.4.26.9 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.

2.4.26.10 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.

2.4.26.11 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.

2.4.26.12 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.

2.4.26.13 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.

2.4.26.14 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.

2.4.26.15 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.

2.4.26.16 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.

2.4.26.17 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.

2.4.26.18 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.

2.4.26.19 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.

2.4.26.20 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

2.4.26.21 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.

2.4.26.22 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

2.4.26.23 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.

2.4.26.24 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

2.4.26.25 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.

2.4.26.26 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.

2.4.26.27 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 **IDeckLinkDeckControl StatusCallback::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 **IDeckLink DeckControl::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.

2.4.26.28 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 **IDeckLinkDeckControl StatusCallback::DeckControlEventReceived** callback. Reception of that event signals that the serial timecodes attached to the **IDeckLinkVideoFrame** objects (received via **IDeckLink InputCallback::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 **IDeckLinkDeckControlStatusCallback::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.

2.4.26.29 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.

2.4.26.30 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

2.4.26.31 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.

2.4.26.32 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.

2.4.26.33 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

2.4.27 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
`IDeckLink DeckControl`	`IID_IDeckLinkDeck Control`	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.

2.4.27.1 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

2.4.27.2 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

2.4.27.3 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

2.4.27.4 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 Common Data Types

2.5.1 Basic Types

`boolean`

boolean is represented differently on each platform by using its system type:

Windows	BOOL
Mac OS X	bool
Linux	bool

`Strings`

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

Windows	BSTR
Mac OS X	CFStringRef
Linux	char *

2.5.2 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.

2.5.3 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	25	1
bmdModePALp	720	576	50	1	50	1
bmdModeHD720p50	1280	720	50	1	50	1
bmdModeHD720p5994	1280	720	60/1.001	1	60000	1001
bmdModeHD720p60	1280	720	60	1	60	1
bmdModeHD1080p2398	1920	1080	24/1.001	1	24000	1001
bmdModeHD1080p24	1920	1080	24	1	24	1
bmdModeHD1080p25	1920	1080	25	1	25000	1000
bmdModeHD1080p2997	1920	1080	30/1.001	1	30000	1001
bmdModeHD1080p30	1920	1080	30	1	30000	1000

continued over page...

Mode	Width	Height	Frames per Second	Fields per Frame	Suggested Time Scale	Display Duration
bmdModeHD1080i50	1920	1080	25	2	25	1
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	2	24000	1001
bmdMode2k24	2048	1556	24	2	24	1
bmdMode2k25	2048	1556	25	2	25000	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.

2.5.4 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 into 16 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.

2.5.5 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)

2.5.6 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

2.5.7 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.

2.5.8 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::SetAncillaryData**
bmdVideoOutputDualStream3D	Set the DeckLink device to output the 3D version of the selected **BMDDisplayMode** display mode.

2.5.9 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".

2.5.10 Frame preview format

BMD3DPreviewFormat enumerates the dual preview formats available for the DeckLink screen preview. The preview format can be set using **IDeckLinkGLScreenPreviewHelper::SetLinkedFramePreviewFormat**.

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.

2.5.11 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

2.5.12 Audio Sample Rates

BMDAudioSampleRate enumerates the possible audio sample rates.

bmdAudioSampleRate48kHz	48 kHz sample rate

2.5.13 Audio Sample Types

BMDAudioSampleType enumerates the possible audio sample types.

bmdAudioSampleType16bitInteger	16 bit audio sample
bmdAudioSampleType32bitInteger	32 bit audio sample

2.5.14 DeckLink Information ID

BMDDeckLinkAPIInformationID enumerates a set of information details which may be queried (see **IDeckLinkAPIInformation** Interface for details).

Name	Type	Description
BMDDeckLinkAPIVersion	String	The user viewable API version number. This allocated string must be freed by the caller when no longer required.
BMDDeckLinkAPIVersion	Int	The API version number. Format: Word Decreasing Address Order **Byte 4**: Major Version; **Byte 3**: Minor Version; **Byte 2**: Sub Version; **Byte 1**: Extra

2.5.15 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.
BMDDeckLinkSerialDevicePortName	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 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. 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.
BMDDeckLinkVideoInputConnections	Int	The video input connections supported by the hardware (see **BMDVideoConnection** 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.
BMDDeckLinkDeviceBusyState	Int	The current busy state of the device. (See **BMDDeviceBusyState** for more information)

2.5.16 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
bmdDeckLinkConfigUse1080pNotPsF	Flag	In 1080 modes use P not PsF if this setting is enabled.
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.
bmdDeckLinkConfig3GBpsVideoOutput	Flag	True if 3GB/s video output is enabled.
bmdDeckLinkConfigBlackVideoOutputDuringCapture	Flag	True if black output during capture is enabled. This feature is only supported on legacy DeckLink devices.
bmdDeckLinkConfigLowLatencyVideoOutput	Flag	If set true, the latency in scheduled video playback will be improved. If set true, **DisplayVideoFrameSync** will guarantee that the provided frame will be output as soon the previous frame output has been completed. This feature should be considered experimental.

Name	Type	Description
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.
bmdDeckLinkConfigReferenceInputTimingOffset	Int(64)	Adjust genlock timing offset. The supported range is between -511 and 511.
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.
bmdDeckLinkConfig32PulldownSequenceInitialTimecodeFrame	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.
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.

Name	Type	Description
bmdDeckLinkConfigVideoOutputComponentLumaGain	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.
bmdDeckLinkConfigVideoOutputComponentChromaBlueGain	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.
bmdDeckLinkConfigVideoOutputComponentChromaRedGain	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.
bmdDeckLinkConfigVideoOutputCompositeLumaGain	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.
bmdDeckLinkConfigVideoOutputCompositeChromaGain	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.
bmdDeckLinkConfigVideoOutputSVideoLumaGain	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.
bmdDeckLinkConfigVideoOutputSVideoChromaGain	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.
bmdDeckLinkConfigVideoInputComponentLumaGain	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.
bmdDeckLinkConfigVideoInputComponentChromaBlueGain	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.

Name	Type	Description
bmdDeckLinkConfigVideoInputComponentChromaRedGain	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.
bmdDeckLinkConfigVideoInputCompositeLumaGain	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.
bmdDeckLinkConfigVideoInputCompositeChromaGain	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.
bmdDeckLinkConfigVideoInputSVideoLumaGain	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.
bmdDeckLinkConfigVideoInputSVideoChromaGain	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.

2.5.17 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.

2.5.18 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.

2.5.19 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

2.5.20 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.

2.5.21 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

2.5.22 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

2.5.23 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))

2.5.24 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.

2.5.25 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.

2.5.26 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.

2.5.27 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

2.5.28 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.
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.

2.5.29 BMDTimecodeFlags

BMDTimecodeFlags enumerates the possible flags that accompany a timecode.

bmdTimecodeFlagDefault	timecode is a non-drop timecode
bmdTimecodeIsDropFrame	timecode is a drop timecode

2.5.30 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

2.5.31 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

2.5.32 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**.

2.5.33 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.

2.5.34 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).

2.5.35 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.

2.5.36 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

2.5.37 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.

2.5.38 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.

2.5.39 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.

2.6 Formulas

2.6.1 Color space Conversion Formulas

The following basic equations convert between gamma-corrected RGB and YUV colour spaces.

SD RGB to YUV conversion

Y = 0.257R + 0.504G + 0.098B + 16
Cb = -0.148R - 0.291G + 0.439B + 128
Cr = 0.439R - 0.368G - 0.071B + 128

HD RGB to YUV conversion

Y = 0.183R + 0.614G + 0.062B + 16
Cb = -0.101R - 0.338G + 0.439B + 128
Cr = 0.439R - 0.399G - 0.040B + 128

SD YUV to RGB conversion

R = 1.164(Y - 16) + 1.596 (Cr - 128)
G = 1.164(Y - 16) - 0.813 (Cr - 128) - 0.391 (Cb - 128)
B = 1.164(Y - 16) + 2.018(Cb - 128)

HD YUV to RGB conversion

R = 1.164(Y - 16) + 1.793(Cr - 128)
G = 1.164(Y - 16) - 0.534(Cr - 128) - 0.213(Cb - 128)
B = 1.164(Y - 16) + 2.115(Cb - 128)

Ensure that output values are clamped to keep them in the 0 - 255 range. The valid SMPTE ranges are Y: 16-240, Cb/ Cr: 16-235.
IDeckLinkOutput::DisplayVideoFrameSync or **IDeckLinkOutput::ScheduleVideoFrame** can convert RGB formatted frames to YUV for output.

www.blackmagic-design.com

SDK Software Developers Kit