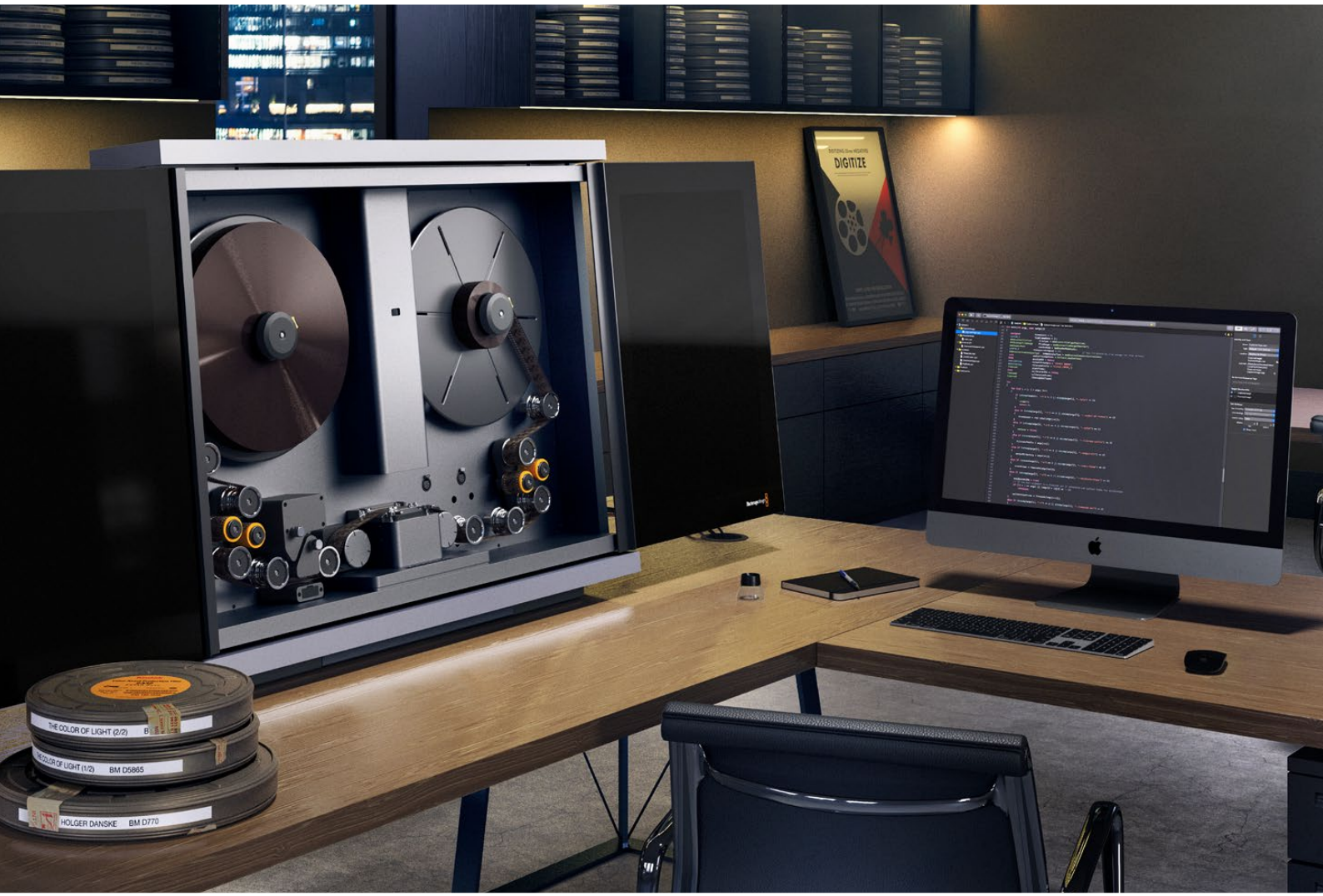
April 2024

Software Developers Kit

Blackmagicdesign

SDK-Cintel Scanner



Contents

Introduction 8

Welcome 8

Overview 8

API Design 9

API Design 9

Supported Products 9

Supported Operating Systems 9

Object Interfaces 9

Reference Counting 9

Interface Stability 9

New Interfaces 10

Updated Interfaces 10

Deprecated Interfaces 10

Removed Interfaces 10

Interface Reference 10

IUnknown Interface 10

IUnknown::QueryInterface method 11

IUnknown::AddRef method 11

IUnknown::Release method 11

Section 1 — Scanner API 12

1.1 Using the Scanner API in a project 12

1.2 Accessing Scanner devices 12

1.3 High level interface 12

1.3.1 Capture 13

1.3.2 Scanner Device Notification 13

1.3.3 Cintel Metadata 13

Section 2 — Interface Reference 14

2.1 IScannerAPIInformation Interface 14

2.1.1 IScannerAPIInformation::GetFlag method 14

2.1.2 IScannerAPIInformation::GetInt method 15

2.1.3 IScannerAPIInformation::GetFloat method 15

2.1.4 IScannerAPIInformation::GetString method 16

2.2 IScannerDiscovery Interface 16

2.2.1 IScannerDiscovery::InstallDeviceNotifications method 17

2.2.2 IScannerDiscovery::UninstallDeviceNotifications method 17

2.3 IScannerDeviceNotificationCallback Interface 17

2.3.1 IScannerDeviceNotificationCallback::ScannerDeviceArrived method 18

2.3.2	IScannerDeviceNotificationCallback::ScannerDeviceRemoved method	18
2.4	IScanner Interface	19
2.4.1	IScanner::InstallScannerNotifications method	19
2.4.2	IScanner::UninstallScannerNotifications method	20
2.4.3	IScanner::GetModelName method	20
2.4.4	IScanner::GetDisplayName method	21
2.4.5	IScanner::GetTemperature method	21
2.4.6	IScanner::GetDeckLinkDevice method	22
2.5	IScannerNotifications Interface	22
2.5.1	IScannerNotifications::SystemStateChanged method	23
2.5.2	IScannerNotifications::MessageEvent method	23
2.6	IScannerInputFrame Interface	24
2.6.1	IScannerInputFrame::GetMetadataInt method	25
2.6.2	IScannerInputFrame::GetMetadataFloat method	25
2.6.3	IScannerInputFrame::GetMetadataFlag method	26
2.6.4	IScannerInputFrame::GetMetadataString method	26
2.6.5	IScannerInputFrame::HasVideo method	27
2.6.6	IScannerInputFrame::GetWidth method	27
2.6.7	IScannerInputFrame::GetHeight method	27
2.6.8	IScannerInputFrame::GetRowBytes method	28
2.6.9	IScannerInputFrame::GetPixelFormat method	28
2.6.10	IScannerInputFrame::GetFlags method	28
2.6.11	IScannerInputFrame::GetBytes method	28
2.6.12	IScannerInputFrame::GetTimecode method	29
2.6.13	IScannerInputFrame::GetDisplayMode method	29
2.6.14	IScannerInputFrame::GetStreamTime method	29
2.6.15	IScannerInputFrame::GetHardwareReferenceTimestamp method	30
2.6.16	IScannerInputFrame::HasAudio method	30
2.6.17	IScannerInputFrame::GetAudioSampleFrameCount method	30
2.6.18	IScannerInputFrame::GetAudioBytes method	31
2.7	IScannerInput Interface	31
2.7.1	IScannerInput::DoesSupportVideoMode method	32
2.7.2	IScannerInput::EnableVideoInput method	32
2.7.3	IScannerInput::DisableVideoInput method	33
2.7.4	IScannerInput::GetAvailableVideoFrameCount method	33
2.7.5	IScannerInput::EnableAudioInput method	33
2.7.6	IScannerInput::DisableAudioInput method	34
2.7.7	IScannerInput::GetAvailableAudioSampleFrameCount method	34
2.7.8	IScannerInput::StartStreams method	34
2.7.9	IScannerInput::StopStreams method	35
2.7.10	IScannerInput::PauseStreams method	35

2.7.11	IScannerInput::FlushStreams method	35
2.7.12	IScannerInput::SetCallback method	36
2.7.13	IScannerInput::GetHardwareReferenceClock method	36
2.8	IScannerInputCallback Interface	37
2.8.1	IScannerInputCallback::ScannerInputFrameArrived method	37
2.8.2	IScannerInputCallback::ScannerInputFormatChanged method	38
2.9	IScannerFilmConfiguration Interface	38
2.9.1	IScannerFilmConfiguration::SetFilmType method	39
2.9.2	IScannerFilmConfiguration::GetFilmType method	40
2.9.3	IScannerFilmConfiguration::GetDefaultFilmType method	40
2.9.4	IScannerFilmConfiguration::GetFilmGaugeAvailable method	40
2.9.5	IScannerFilmConfiguration::SetFilmGauge method	41
2.9.6	IScannerFilmConfiguration::GetFilmGauge method	41
2.9.7	IScannerFilmConfiguration::GetDefaultFilmGauge method	42
2.9.8	IScannerFilmConfiguration::SetFilmFrameRate method	42
2.9.9	IScannerFilmConfiguration::GetFilmFrameRate method	43
2.9.10	IScannerFilmConfiguration::GetDefaultFilmFrameRate method	43
2.9.11	IScannerFilmConfiguration::GetMinimumFilmFrameRate method	43
2.9.12	IScannerFilmConfiguration::GetMaximumFilmFrameRate method	44
2.9.13	IScannerFilmConfiguration::SetFeedWind method	44
2.9.14	IScannerFilmConfiguration::GetFeedWind method	45
2.9.15	IScannerFilmConfiguration::GetDefaultFeedWind method	45
2.9.16	IScannerFilmConfiguration::SetTakeupWind method	46
2.9.17	IScannerFilmConfiguration::GetTakeupWind method	46
2.9.18	IScannerFilmConfiguration::GetDefaultTakeupWind method	47
2.9.19	IScannerFilmConfiguration::SetFeedRollType method	47
2.9.20	IScannerFilmConfiguration::GetFeedRollType method	48
2.9.21	IScannerFilmConfiguration::GetDefaultFeedRollType method	48
2.9.22	IScannerFilmConfiguration::SetTakeupRollType method	49
2.9.23	IScannerFilmConfiguration::GetTakeupRollType method	49
2.9.24	IScannerFilmConfiguration::GetDefaultTakeupRollType method	50
2.10	IScannerTransport Interface	50
2.10.1	IScannerTransport::SetRunSpeed method	52
2.10.2	IScannerTransport::GetRunSpeed method	52
2.10.3	IScannerTransport::GetMinimumRunSpeed method	53
2.10.4	IScannerTransport::GetMaximumRunSpeed method	53
2.10.5	IScannerTransport::GetDefaultRunSpeed method	53
2.10.6	IScannerTransport::SetShuttleSpeed method	54
2.10.7	IScannerTransport::GetShuttleSpeed method	54
2.10.8	IScannerTransport::GetMinimumShuttleSpeed method	55
2.10.9	IScannerTransport::GetMaximumShuttleSpeed method	55

2.10.10	IScannerTransport::GetDefaultShuttleSpeed method	55
2.10.11	IScannerTransport::SetAcceleration method	56
2.10.12	IScannerTransport::GetAcceleration method	56
2.10.13	IScannerTransport::GetMinimumAcceleration method	57
2.10.14	IScannerTransport::GetMaximumAcceleration method	57
2.10.15	IScannerTransport::GetDefaultAcceleration method	58
2.10.16	IScannerTransport::SetFilmTension method	58
2.10.17	IScannerTransport::GetFilmTension method	59
2.10.18	IScannerTransport::GetMinimumFilmTension method	59
2.10.19	IScannerTransport::GetMaximumFilmTension method	60
2.10.20	IScannerTransport::GetDefaultFilmTension method	60
2.10.21	IScannerTransport::SetTimecode method	61
2.10.22	IScannerTransport::GetTimecode method	61
2.10.23	IScannerTransport::SetCue method	62
2.10.24	IScannerTransport::SendCommand method	62
2.10.25	IScannerTransport::SetCaptureMode method	63
2.10.26	IScannerTransport::GetCaptureMode method	63
2.10.27	IScannerTransport::GetDefaultCaptureMode method	64
2.10.28	IScannerTransport::GetFeedFramesRemaining method	64
2.10.29	IScannerTransport::GetTakeupFramesRemaining method	65
2.10.30	IScannerTransport::GetPreroll method	65
2.10.31	IScannerTransport::SetAdvancedFlags method	66
2.10.32	IScannerTransport::GetAdvancedFlags method	66
2.10.33	IScannerTransport::GetHDRCaptureAvailable method	67
2.11	IScannerColorProperties Interface	67
2.11.1	IScannerColorProperties::SetLEDIllumination method	68
2.11.2	IScannerColorProperties::GetLEDIllumination method	69
2.11.3	IScannerColorProperties::GetDefaultLEDIllumination method	69
2.11.4	IScannerColorProperties::StartFixedPatternGainCalibration method	70
2.11.5	IScannerColorProperties::GetFixedPatternGainCalibrated method	70
2.11.6	IScannerColorProperties::StartLEDCalibration method	70
2.11.7	IScannerColorProperties::GetLEDCalibrationStatus method	71
2.11.8	IScannerColorProperties::GetHDRCalibrationStatus method	71
2.11.9	IScannerColorProperties::GetFixedPatternGainCalibrationStatus method	72
2.11.10	IScannerColorProperties::SetFocusPeakingEnable method	72
2.11.11	IScannerColorProperties::GetFocusPeakingEnable method	73
2.11.12	IScannerColorProperties::GetDefaultFocusPeakingEnable method	73
2.11.13	IScannerColorProperties::SetEnableHDRMode method	73
2.11.14	IScannerColorProperties::GetEnableHDRMode method	74
2.11.15	IScannerColorProperties::GetDefaultEnableHDRMode method	74
2.12	IScannerImageStabilizer Interface	75

2.12.1	IScannerImageStabilizer::SetStabilizationEnable method	76
2.12.2	IScannerImageStabilizer::GetStabilizationEnable method	76
2.12.3	IScannerImageStabilizer::GetDefaultStabilizationEnable method	77
2.12.4	IScannerImageStabilizer::GetDefaultStabilizationEnableXY method	77
2.12.5	IScannerImageStabilizer::GetStabilizationEnableXY method	78
2.12.6	IScannerImageStabilizer::GetDefaultStabilizationEnableXY method	78
2.12.7	IScannerImageStabilizer::SetStabilizationViewRegion method	79
2.12.8	IScannerImageStabilizer::GetStabilizationViewRegion method	79
2.12.9	IScannerImageStabilizer::GetDefaultStabilizationViewRegion method	79
2.12.10	IScannerImageStabilizer::SetStabilizationOffsetX method	80
2.12.11	IScannerImageStabilizer::GetStabilizationOffsetX method	80
2.12.12	IScannerImageStabilizer::GetMinimumStabilizationOffsetX method	80
2.12.13	IScannerImageStabilizer::GetMaximumStabilizationOffsetX method	81
2.12.14	IScannerImageStabilizer::GetDefaultStabilizationOffsetX method	81
2.13	IScannerCapture Interface	82
2.13.1	IScannerCapture::SetCompressionType method	82
2.13.2	IScannerCapture::GetCompressionType method	83
2.13.3	IScannerCapture::GetDefaultCompressionType method	83
2.14	IScannerAudioSelect Interface	84
2.14.1	IScannerAudioSelect::SetAudioSelectSource method	84
2.14.2	IScannerAudioSelect::GetAudioSelectSource method	85
2.14.3	IScannerAudioSelect::GetDefaultAudioSelectSource method	85
2.14.4	IScannerAudioSelect::GetAudioSourceAvailable method	86
2.14.5	IScannerAudioSelect::SetExternalAudioFormat method	86
2.14.6	IScannerAudioSelect::GetExternalAudioFormat method	87
2.14.7	IScannerAudioSelect::GetDefaultExternalAudioFormat method	87
2.14.8	IScannerAudioSelect::SetSyncConfiguration method	88
2.14.9	IScannerAudioSelect::GetSyncConfiguration method	88
2.14.10	IScannerAudioSelect::GetDefaultSyncConfiguration method	89
2.15	IScannerReader Interface	89
2.15.1	IScannerReader::GetReaderPresent method	90
2.15.2	IScannerReader::GetReaderMode method	90
2.15.3	IScannerReader::GetDefaultReaderMode method	91
2.15.4	IScannerReader::SetReaderMode method	91
2.15.5	IScannerReader::GetMagneticAudioAllowed method	92
2.15.6	IScannerReader::SetAudioMode method	92
2.15.7	IScannerReader::GetAudioMode method	92
2.15.8	IScannerReader::GetDefaultAudioMode method	93
2.15.9	IScannerReader::GetAudioPresent method	93

Section 3 — Common Data Types 94

3.1	Basic Types	94
3.2	Cintel Scanner API Information ID	95
3.3	Display Modes	95
3.4	Pixel Formats	95
3.5	Frame Flags	96
3.6	Audio Sample Rate	96
3.7	Audio Sample Types	96
3.8	BMDScannerAudioSelectSource	96
3.9	BMDScannerExternalAudioFormat	97
3.10	BMDScannerTimecodeBCD	97
3.11	BMDScannerTimeValue	97
3.12	BMDScannerTimeScale	97
3.13	Frame Metadata ID	98
3.14	BMDScannerTransportCommand	99
3.14.1	BMDScannerFilmType	100
3.14.2	BMDScannerFilmGauge	100
3.14.3	BMDScannerWindType	100
3.14.4	BMDScannerCompressionType	100
3.14.5	BMDScannerRollType	101
3.14.6	BMDScannerState	101
3.14.7	BMDScannerMessage	102
3.14.8	BMDScannerMessageSeverity	103
3.14.9	BMDScannerCalibrationStatus	103
3.15	BMDReaderMode	103
3.16	BMDReaderAudioMode	104
3.17	BMDScannerCaptureMode	104
3.18	BMDScannerAdvancedFlag	104
3.19	BMDScannerSyncLTCConfig	104

Introduction

Welcome

Thank you for downloading the Blackmagic Design Cintel Scanner Software Developers Kit.

Overview

The Cintel Scanner Software Developer Kit provides a stable, cross-platform interface to help developers create applications to control all aspects of your scanner.

Functions are provided so you can write software to manage scanner settings, acquisition, metadata, processing, and file management. For example, you could make your own scanner workflow software with automated verifications, or to export CRI images to a custom codec. Software you create with the Cintel Scanner SDK can be used instead of DaVinci Resolve to control your scanner, which may be useful for integration with third party systems, automation, or if the host computer has limited memory or CPU resources.

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 and two sample applications to demonstrate how to use key functions of the scanner hardware with the SDK. The 'CaptureImage' example application and source code controls the scanner, image capture, and writing CRI and associated audio files to disk. The source code for CaptureImage is provided to you can see how easy it is to create an application to scan film and create CRI files, and to inspire you to develop customized applications. The 'ProcessImage' example application loads and uses the provided ProcessImage library to demonstrate the processing steps required to create images of equivalent quality as DaVinci Resolve. It is important to note that captures done through the SDK can easily be imported into DaVinci Resolve, and clips from DaVinci Resolve can easily be processed with the SDK. CaptureImage and ProcessImage also support Cintel HDR capture and processing.

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

The libraries supporting this SDK are included in the Cintel Scanner SDK package, which is separate to the scanner software installer. Applications built against the interfaces shipped in the SDK dynamically link against the library installed on the host computer for your scanner.

The SDK interface is modeled on Microsoft's Component Object Model (COM). On Microsoft Windows platforms, it is provided as a native COM interface registered with the operating system. On other platforms, application code is provided to allow the same COM style interface to be used. The COM model provides a paradigm for creating flexible and extensible interfaces with minimal overhead.

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

The Blackmagic Design Developer website provides video tutorials and FAQs for developing software for your scanner.

Please visit at **www.blackmagicdesign.com/developer**

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

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

API Design

API Design

Supported Products

The Cintel Scanner SDK provides programmatic access to all features of your scanner.
The term 'Cintel Scanner' is used as a generic term to refer to all models of Cintel Scanner.

Supported Operating Systems

This SDK is supported on Mac, Windows and Linux operating systems. The release notes supplied with the scanner package includes details of supported operating system versions.

Object Interfaces

The API provides high-level interfaces to allow scanning, debayering, and playback of video and audio from a range of 16mm and 35mm film, as well as low-level interfaces for controlling features of your scanner.

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' (IID).
On platforms with native COM support, you can use an IID 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.

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.

Interface Stability

The SDK provides a set of stable interfaces for accessing Blackmagic Design hardware. While 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.

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** 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 reduce the functionality of their application when an interface is not available, or to notify the user that they must install a newer version of the Blackmagic drivers.

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. The replacement interface will have a new identifier and remain as similar to the original as possible.

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.

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.

Interface Reference

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

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 with no impact on 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.

IUnknown::QueryInterface method

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

Syntax

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

Parameters

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

Return Values

Value	Description
E_NOINTERFACE	Interface was not found.
S_OK	Success.

IUnknown::AddRef method

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

Syntax

```
ULONG       AddRef();
```

Return Values

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

IUnknown::Release method

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

Syntax

```
ULONG       Release();
```

Return Values

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

Section 1 — Scanner API

1.1 Using the Scanner API in a project

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

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

Windows	Scanner X.Y/Win\Include/ScannerAPI.idl
Mac	Scanner X.Y/Mac/Include/ScannerAPI.h Scanner X.Y/Mac/Include/ScannerAPIDispatch.cpp
Linux	Scanner X.Y/Linux/Include/ScannerAPI.h Scanner X.Y/Linux/Include/ScannerAPIDispatch.cpp

1.2 Accessing Scanner devices

The main entry point to the Scanner API is through the **IScannerDiscovery** interface. For more information, see the '**IScannerDiscovery** Interface' section.

```
COMPtr<IScannerDiscovery> m_scannerDiscovery;

m_scannerDiscovery.Attach(CreateScannerDiscoveryInstance());

m_scannerDiscovery->InstallDeviceNotifications(this);

m_scannerDiscovery->UninstallDeviceNotifications();
```

After that, notifications for **ScannerDeviceArrived(IScanner* scanner)** and **ScannerDeviceRemoved(IScanner* scanner)** should be received when a scanner is connected. Then you can use the **IScanner** interface provided to Query the relevant interface as shown here for the **IScannerInput** interface:

```
COMPtr<IScannerInput> scannerInput;

scanner->QueryInterface(IID_IScannerInput, (void **)scannerInput.outArg());
```

1.3 High level interface

The Scanner API uses the Desktop Video driver, however it is fully abstracted so the user of the Scanner API does not require any knowledge of the DeckLink API. This 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.

1.3.1 Capture

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

- **IScannerInput::EnableVideoInput**
- **IScannerInput::EnableAudioInput**
- **IScannerInput::StartStreams**

  While streams are running:
  – receive calls to **IScannerInputCallback::ScannerInputFrameArrived** with video frame and corresponding audio packet.
  – receive calls to **IScannerInputCallback::ScannerInputFormatChanged** if the display mode changes.
- **IScannerInput::StopStreams**

Audio may be 'pulled' from a separate thread if desired.

If audio is not required, the call to **IScannerInput::EnableAudioInput** may be omitted and the **IScannerInputCallback::ScannerInputFrameArrived** callback will receive NULL audio packets.

1.3.2 Scanner Device Notification

A callback notification can be provided to an application when a Thunderbolt or PCIe based scanner is connected or disconnected. For more information, see the 'Accessing Scanner Devices' section.

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

- Create a callback class that subclasses **IScannerNotifications** and implements all of its methods. The callback class will be called asynchronously from an API private thread. Create an instance of the callback class.
- Call **IScanner::InstallScannerNotifications** and provide the **IScannerNotificationCallBack** object.
- **IScannerNotifications::SystemStateChanged** is called whenever the Scanner's internal state changes.
- **IScannerNotifications::MessageEvent** is called if the Scanner needs to notify of any particular events.
- Before the application exits, call **IScannerNotifications::UninstallScannerNotifications**.

1.3.3 Cintel Metadata

Metadata capture is supported by Cintel devices. To access this metadata, simply use the GetMetadataInt, Float, Flag and String functions when an IScannerInputFrame* is received via ScannerInputFrameArrived. The enumeration BMDScannerFrameMetadataID supports these queries.

Section 2 — Interface Reference

2.1 IScannerAPIInformation Interface

The **IScannerAPIInformation** object interface provides global information about the Scanner API installation such as configuration information. A reference to an **IScannerAPIInformation** object interface may be obtained from **CreateScannerAPIInformationInstance**.

Public Member Functions	
Method	**Description**
GetFlag	Get the boolean value of a configuration flag associated with a specified **BMDScannerAPIInformationID**.
GetInt	Get the integer value of a configuration flag associated with a specified **BMDScannerAPIInformationID**.
GetFloat	Get the floating point value of a configuration flag associated with a specified **BMDScannerAPIInformationID**.
GetString	Get the string the value of a configuration flag associated with a specified **BMDScannerAPIInformationID**.

2.1.1 IScannerAPIInformation::GetFlag method

The **GetFlag** method returns the boolean value for a given **BMDScannerAPIInformationID**.

Syntax

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

Parameters

Name	**Direction**	**Description**
`cfgID`	in	**BMDScannerAPIInformationID** 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**.

2.1.2 IScannerAPIInformation::GetInt method

The **GetInt** method returns the 64 bit integer value for a given **BMDScannerAPIInformationID**.

Syntax

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

Parameters

Name	Direction	Description
`value`	out	Value of integer corresponding to **cfgID**.
`cfgID`	in	**BMDScannerAPIInformationID** to get integer value.

Return Values

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

2.1.3 IScannerAPIInformation::GetFloat method

The **GetFloat** method returns the floating point value for a given **BMDScannerAPIInformationID**.

Syntax

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

Parameters

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

Return Values

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

2.1.4 IScannerAPIInformation::GetString method

The **GetString** method returns the string value for a given **BMDScannerAPIInformationID**.

Syntax

```
HRESULT    GetString (BMDScannerAPIInformationID cfgID, const char **value);
```

Parameters

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

Return Values

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

2.2 IScannerDiscovery Interface

The **IScannerDiscovery** object interface is used to install or remove the callback for receiving Scanner device discovery notifications.

A reference to an **IScannerDiscovery** object interface may be obtained from **CreateScannerDiscoveryInstance**.

Related Interfaces

Interface	Interface ID	Description
`IScannerDevice NotificationCallback`	`IID_IScannerDevice NotificationCallback`	A device notification callback can be installed with IScannerDiscovery:: InstallDeviceNotifications or uninstalled with IScannerDiscovery:: UninstallDeviceNotifications

Public Member Functions	
Method	**Description**
InstallDeviceNotifications	Install Scanner device notifications callback.
UninstallDeviceNotifications	Remove Scanner device notifications callback.

2.2.1 IScannerDiscovery::InstallDeviceNotifications method

The **InstallDeviceNotifications** method installs the **IScannerDeviceNotificationCallback** callback, which will be called when a new Scanner device becomes available.

Syntax

```
HRESULT     InstallDeviceNotifications (IScannerDeviceNotificationCallback
            *deviceNotificationCallback);
```

Parameters

Name	Direction	Description
`deviceNotificationCallback`	in	Callback object implementing the IScannerDeviceNotificationCallback object interface.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	The parameter variable is NULL.

2.2.2 IScannerDiscovery::UninstallDeviceNotifications method

The **UninstallDeviceNotifications** method removes the Scanner device notifications callback. When this method returns, it guarantees there are no ongoing callbacks to the **IScannerNotificationCallback** instance.

Syntax

```
HRESULT     UninstallDeviceNotifications (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.3 IScannerDeviceNotificationCallback Interface

The **IScannerDeviceNotificationCallback** object interface manages device arrival and removal notification callbacks.

Public Member Functions	
Method	**Description**
ScannerDeviceArrived	A scanner device has arrived.
ScannerDeviceRemoved	A scanner device has been removed.

2.3.1 IScannerDeviceNotificationCallback::ScannerDeviceArrived method

The **ScannerDeviceArrived** method is called when a new scanner becomes available. This method will be called on an API private thread.

This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT    ScannerDeviceArrived (IScanner *scannerDevice);
```

Parameters

Name	Direction	Description
`scannerDevice`	out	Scanner device. The **IScanner** reference will be released when the callback returns. To hold on to it beyond the callback, call **AddRef** or assign it to a new **COMPtr**. Your application then owns the **IScanner** reference and is responsible for managing the **IScanner** object's lifetime. The reference can be released at any time (including in the **ScannerDeviceRemoved** callback) by calling **Release**, or by having used a **COMPtr** and simply assigning it to **nullptr**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.3.2 IScannerDeviceNotificationCallback::ScannerDeviceRemoved method

The **ScannerDeviceRemoved** method is called when a Scanner device is disconnected.

This method will be called on an API private thread.

This method is abstract in the base interface and must be implemented by the application developer. The result parameter (required by COM) is ignored by the caller.

Syntax

```
HRESULT    ScannerDeviceRemoved (IScanner *scannerDevice);
```

Parameters

Name	Direction	Description
`scannerDevice`	in	Scanner device.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4 IScanner Interface

The **IScanner** object interface represents a physical Scanner device. A reference to it is received when **IScannerDeviceNotificationCallback::ScannerDeviceArrived** is called.

Related Interfaces

Interface	Interface ID	Description
`IScannerDevice NotificationCallback`	`IID_IScannerDevice NotificationCallback`	IScannerDeviceNotificationCallback:: ScannerDeviceArrived or ScannerDeviceRemoved informs the reference to the IScanner being added or removed.

Public Member Functions	
Method	**Description**
InstallScannerNotifications	Install Scanner device notifications callback.
UninstallScannerNotifications	Remove Scanner device notifications callback.
GetModelName	Method to get Scanner device model name.
GetDisplayName	Method to get a device name suitable for user interfaces.
GetTemperature	Method to get the current sensor temperature.
GetDeckLinkDevice	Returns the IDeckLink reference representing the IScanner device, compatible with the DeckLink SDK, should any features of the DeckLink SDK be required for use with your Scanner.

2.4.1 IScanner::InstallScannerNotifications method

The **InstallScannerNotifications** method installs **IScannerNotifications** callbacks.

Syntax

```
HRESULT     InstallScannerNotifications (IScannerNotifications
                                         *scannerNotificationCallback);
```

Parameters

Name	Direction	Description
`scannerNotificationCallback`	in	IScannerNotifications interface reference.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.2 IScanner::UninstallScannerNotifications method

The **UninstallScannerNotifications** method uninstalls **IScannerNotifications** callbacks.

Syntax

```
HRESULT    UninstallScannerNotifications (IScannerNotifications
                                          *scannerNotificationCallback);
```

Parameters

Name	Direction	Description
`scannerNotificationCallback`	in	IScannerNotifications interface reference

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.3 IScanner::GetModelName method

The **GetModelName** method can be used to get the Scanner 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

2.4.4 IScanner::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	Failure
S_OK	Success

2.4.5 IScanner::GetTemperature method

The **GetTemperature** method returns the current sensor ttemperature. It's important to note that the temperature of the film scanning sensor inside Cintel Scanner and Cintel Scanner 2 is expected to be 44 degrees celsius.

Syntax

```
HRESULT    GetTemperature (uint32_t *temperature);
```

Parameters

Name	Direction	Description
temperature	out	Sensor temperature in degrees celsius.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.4.6 IScanner::GetDeckLinkDevice method

The **GetDeckLinkDevice** method can be used to get the **IDeckLink** ireference representing the **IScanner** interface if you want to use any features of the DeckLink SDK with your Scanner.

Syntax

```
HRESULT    GetDeckLinkDevice (IUnknown **deckLinkDevice);
```

Parameters

Name	Direction	Description
deckLinkDevice	out	COMPtr to place the IDeckLink reference into.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.5 IScannerNotifications Interface

The **IScannerNotifications** interface manages scanner system state change and error notifications.

Extend the **IScannerNotifications** interface from the object reference which was passed to **IScanner::InstallScannerNotifications**, after which Scanner specific notifications will also be received, such as SystemStateChanged, which contains Scanner specific states such as Loaded or Unloaded.

Related Interfaces

Interface	Interface ID	Description
IScanner	IID_IScanner	An IScannerNotifications interface can be extended once the instantiating object is passed to IScanner:: InstallScannerNotifications(this)

Public Member Functions	
Method	**Description**
SystemStateChanged	Extend this function to receive Scanner specific messages.
MessageEvent	Extend this function to receive Scanner specific state machine messages.

2.5.1 IScannerNotifications::SystemStateChanged method

The **SystemStateChanged** method informs the user of **BMDScannerState** changes, such as Loaded or Unloaded.

Syntax

```
HRESULT    SystemStateChanged (BMDScannerState state);
```

Parameters

Name	Direction	Description
state	in	The latest state of the scanner.

Return Values

Value	Description
S_OK	Success

2.5.2 IScannerNotifications::MessageEvent method

The **MessageEvent** method informs the user of a variety of important messages, for examples errors and warnings.

Syntax

```
HRESULT    MessageEvent (BMDScannerMessage eventCode,
                         BMDScannerMessageSeverity severity,
                         uint32_t numParams, uint64_t *params);
```

Parameters

Name	Direction	Description
eventCode	in	**BMDScannerMessage** containing the message descriptor.
severity	in	**BMDScannerMessageSeverity** containing the severity of the message
numParams	in	Number of associated parameters, if any. If it is zero, params can be a NULL pointer.
params	in	See **BMDScannerMessage** enumeration comments for any potentially associated parameter data for a given **eventCode**

Return Values

Value	Description
S_OK	Success

2.6 IScannerInputFrame Interface

The **IScannerInputFrame** object interface represents a video input frame.

An **IScannerVideoFrame** object interface with the **bmdScannerFrameContainsCintelMetadata** flag may use this interface to query the metadata parameters associated with the video frame.

Related Interfaces

Interface	Interface ID	Description
`IScannerInput`	`IID_IScannerInput`	New input frames are returned to **IScannerInputCallback::ScannerInputFrameArrived** by the **IScannerInput** interface.

Public Member Functions	
Method	**Description**
GetMetadataInt	Get the current integer value of a metadata item associated with the given **BMDScannerFrameMetadataID**.
GetMetadataFloat	Get the current floating point value of a metadata item associated with the given **BMDScannerFrameMetadataID**.
GetMetadataFlag	Get the current boolean value of a metadata item associated with the given **BMDScannerFrameMetadataID**.
GetMetadataString	Get the current string value of a metadata item associated with the given **BMDScannerFrameMetadataID**.
HasVideo	Get whether the frame has video.
GetWidth	Get video frame width in pixels.
GetHeight	Get video frame height in pixels.
GetRowBytes	Get bytes per row for video frame.
GetPixelFormat	Get pixel format for video frame.
GetFlags	Get frame flags.
GetBytes	Get pointer to frame data.
GetTimecode	Get timecode information.
GetDisplayMode	Get the current **BMDScannerDisplayMode**.
GetStreamTime	Return frame time and frame duration based on the time scale.
GetHardwareReferenceTimestamp	Return a timestamp based on frame time and frame duration.
HasAudio	Get whether the frame includes audio data.
GetAudioSampleFrameCount	Get the duration of the audio sample in frames.
GetAudioBytes	Get pointer to audio data.

2.6.1 IScannerInputFrame::GetMetadataInt method

The **GetMetadataInt** method returns the current integer value of a metadata item associated with the given **BMDScannerFrameMetadataID**.

Syntax

```
HRESULT     GetMetadataInt (BMDScannerFrameMetadataID metadataID,
                            int64_t *value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDScannerFrameMetadataID** of the metadata information item.
`value`	out	The integer value corresponding to the metadataID.

Return Values

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

2.6.2 IScannerInputFrame::GetMetadataFloat method

The **GetMetadataFloat** method returns returns the current floating point value of a metadata item associated with the given **BMDScannerFrameMetadataID**.

Syntax

```
HRESULT     GetMetadataFloat (BMDScannerFrameMetadataID metadataID,
                              double *value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDScannerFrameMetadataID** of the metadata information item.
`value`	out	The floating point value corresponding to the metadataID.

Return Values

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

2.6.3 IScannerInputFrame::GetMetadataFlag method

The **GetMetadataFlag** method returns the current boolean value of a metadata item associated with the given **BMDScannerFrameMetadataID**.

Syntax

```
HRESULT     GetMetadataFlag (BMDScannerFrameMetadataID metadataID,
                             bool* value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDScannerFrameMetadataID** of the metadata information item.
`value`	out	The boolean flag corresponding to the metadataID.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no bool type attribute corresponding to metadataID.

2.6.4 IScannerInputFrame::GetMetadataString method

The **GetMetadataString** method returns returns the current string value of a metadata item associated with the given **BMDScannerFrameMetadataID**.

Syntax

```
HRESULT     GetMetadataString (BMDScannerFrameMetadataID metadataID,
                               const char **value);
```

Parameters

Name	Direction	Description
`metadataID`	in	The **BMDScannerFrameMetadataID** of the metadata information item.
`value`	out	The string corresponding to the metadataID.

Return Values

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

2.6.5 IScannerInputFrame::HasVideo method

The **HasVideo** method returns whether the frame has video.

Syntax

```
bool        HasVideo (void);
```

Return Values

Value	Description
true	Video image is present in frame.
false	Video image is not present in frame.

2.6.6 IScannerInputFrame::GetWidth method

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

Note that **GetWidth** uses the full scanner sensor for 35mm and 16mm images, not the region cropped to the film gate. Results are the same for all film formats such as Super 16, or 35mm 3 perf or 4 perf.

For information on how to get the frame dimensions after overscan for perforations and the audio area are removed, see **GetMetadataInt** and the **bmdScannerFrameMetadataCintelImageWidth** metadata flag.

Syntax

```
uint32_t    GetWidth (void);
```

Return Values

Value	Description
Width	Video frame width in pixels.

2.6.7 IScannerInputFrame::GetHeight method

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

Note that **GetHeight** uses the full scanner sensor for 35mm and 16mm images, not the region cropped to the film gate. Results are the same for all film formats such as Super 16, or 35mm 3 perf or 4 perf.

For information on how to get the frame dimensions after overscan for perforations and the audio area are removed, see **GetMetadataInt** and the **bmdScannerFrameMetadataCintelImagetHeight** metadata flag.

Syntax

```
uint32_t    GetHeight (void);
```

Return Values

Value	Description
Height	Video frame height in pixels.

2.6.8 IScannerInputFrame::GetRowBytes method

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

Syntax

```
uint32_t    GetRowBytes (void);
```

Return Values

Value	Description
BytesCount	Number of bytes per row of video frame.

2.6.9 IScannerInputFrame::GetPixelFormat method

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

Syntax

```
BMDScannerPixelFormat      GetPixelFormat (void);
```

Return Values

Value	Description
PixelFormat	Pixel format of video frame (**BMDScannerPixelFormat**).

2.6.10 IScannerInputFrame::GetFlags method

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

Syntax

```
BMDScannerFrameFlags      GetFlags (void);
```

Return Values

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

2.6.11 IScannerInputFrame::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. Valid only while object remains valid.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.12 IScannerInputFrame::GetTimecode method

The **GetTimecode** method returns the timecode specified in the ancillary data for the current frame.

Syntax

BMDScannerTimecodeBCD `GetTimecode (void);`

Return Values

Value	Description
BMDScannerTimecodeBCD	BCD timecode for current frame, or NULL if the timecode is unavailable.

2.6.13 IScannerInputFrame::GetDisplayMode method

The **GetDisplayMode** method returns the current scanner display mode for acquired images. See **BMDScannerDisplayMode**.

Syntax

BMDScannerDisplayMode `GetDisplayMode (void);`

Return Values

Value	Description
Scanner display mode	Scanner display mode.

2.6.14 IScannerInputFrame::GetStreamTime method

The **GetStreamTime** method returns frame time and frame duration based on the time scale.

Syntax

```
HRESULT    GetStreamTime (BMDScannerTimeValue *frameTime, BMDScannerTimeValue
           *frameDuration, BMDScannerTimeScale timeScale);
```

Parameters

Name	Direction	Description
frameTime	out	Frame time in units of **BMDScannerTimeScale**.
frameDuration	out	Frame duration in units of **BMDScannerTimeScale**.
timeScale	in	Time scale for output parameters.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.15 IScannerInputFrame::GetHardwareReferenceTimestamp method

The GetHardwareReferenceTimestamp method returns a timestamp based on frame time and frame duration.

Syntax

```
HRESULT     GetHardwareReferenceTimestamp (BMDScannerTimeScale timeScale,
            BMDScannerTimeValue *frameTime, BMDScannerTimeValue
            *frameDuration);
```

Parameters

Name	Direction	Description
timeScale	in	Time scale for output parameters.
frameTime	out	Frame time in units of **BMDScannerTimeScale**.
frameDuration	out	Frame duration in units of **BMDScannerTimeScale**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.6.16 IScannerInputFrame::HasAudio method

The **HasAudio** method returns whether the frame includes audio data.

Syntax

```
bool        HasAudio (void);
```

Return Values

Value	Description
true	Has audio.
false	Does not have audio.

2.6.17 IScannerInputFrame::GetAudioSampleFrameCount method

The **GetAudioSampleFrameCount** method returns the duration of the audio sample in frames.

Syntax

```
uint32_t    GetAudioSampleFrameCount (void);
```

Return Values

Value	Description
frameCount	Number of audio samples in this frame.

2.6.18 IScannerInputFrame::GetAudioBytes method

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

Syntax

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

Parameters

Name	Direction	Description
buffer	out	Pointer to raw audio buffer. Valid only while object remains valid.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7 IScannerInput Interface

The **IScannerInput** object interface represents the interface for parameters of scanner input, such as video input, audio input, input control and hardware timing.

Related Interfaces

Interface	Interface ID	Description
IScanner	IID_IScanner	Query the **IScannerInput** interface from the IScanner interface. If a firmware update is required, this interface returns **E_ACCESSDENIED** when queried. Run the Cintel Scanner software to update the scanner firmware.

Public Member Functions	
Method	**Description**
DoesSupportVideoMode	Check whether a given video mode is supported for input.
EnableVideoInput	Enable video input.
DisableVideoInput	Disable video input.
GetAvailableVideoFrameCount	The number of available input frames.
EnableAudioInput	Enable audio input with specified sample rate and number of channels.
DisableAudioInput	Disable specified audio input.
GetAvailableAudioSampleFrameCount	Query audio buffer status.
StartStreams	Start encoded capture.
StopStreams	Stop encoded capture.
PauseStreams	Pause encoded capture.
FlushStreams	Remove any buffered video and audio frames.
SetCallback	Register IScannerInputCallback method to call when a frame arrives or the scanner format changes.
GetHardwareReferenceClock	Return hardware timing information for a given timescale.

2.7.1 IScannerInput::DoesSupportVideoMode method

The **DoesSupportVideoMode** method indicates whether a given video mode is supported. Modes may be supported, unsupported or supported with conversion.

Syntax

```
HRESULT     DoesSupportVideoMode (BMDScannerDisplayMode requestedMode,
                                  BMDScannerPixelFormat requestedPixelFormat,
                                  bool *supported);
```

Parameters

Name	Direction	Description
`requestedMode`	in	Video mode to check.
`requestedPixelFormat`	in	Pixel format to check.
`supported`	out	True if video mode is supported, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7.2 IScannerInput::EnableVideoInput method

The **EnableVideoInput** method configures video input and puts the hardware into encoded video capture mode. To start video input, and optionally audio input, call **StartStreams**.

Syntax

```
HRESULT     EnableVideoInput (BMDScannerDisplayMode displayMode,
                              BMDScannerPixelFormat pixelFormat);
```

Parameters

Name	Direction	Description
`displayMode`	in	Video mode to capture.
`pixelFormat`	in	Encoded pixel format to capture.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	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

2.7.3 IScannerInput::DisableVideoInput method

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

Syntax

```
HRESULT     DisableVideoInput (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7.4 IScannerInput::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
E_FAIL	Failure
S_OK	Success

2.7.5 IScannerInput::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 (BMDScannerAudioSampleType sampleType);
```

Parameters

Name	Direction	Description
sampleType	in	Sample type to capture.

Return Values

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

2.7.6 IScannerInput::DisableAudioInput method

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

Syntax

```
HRESULT     DisableAudioInput (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7.7 IScannerInput::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 **IScannerInputCallback** 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

2.7.8 IScannerInput::StartStreams method

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

Syntax

```
HRESULT     StartStreams ();
```

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

2.7.9 IScannerInput::StopStreams method

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

Syntax

```
HRESULT     StopStreams ();
```

Return Values

Value	Description
E_ACCESSDENIED	Input stream has already stopped.
S_OK	Success

2.7.10 IScannerInput::PauseStreams method

The **PauseStreams** method returns 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 ();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7.11 IScannerInput::FlushStreams method

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

Syntax

```
HRESULT     FlushStreams ();
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7.12 IScannerInput::SetCallback method

The **SetCallback** method configures a callback that 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 (IScannerInputCallback *theCallback);
```

Parameters

Name	Direction	Description
theCallback	in	Callback object implementing the **IScannerInputCallback** object interface.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.7.13 IScannerInput::GetHardwareReferenceClock method

The GetHardwareReferenceClock method returns hardware timing information for a given timescale.

Syntax

```
HRESULT     GetHardwareReferenceClock (BMDScannerTimeScale desiredTimeScale,
            BMDScannerTimeValue *hardwareTime, BMDScannerTimeValue
            *timeInFrame, BMDScannerTimeValue *ticksPerFrame);
```

Parameters

Name	Direction	Description
desiredTimeScale	in	Desired time scale.
hardwareTime	out	Hardware reference time in units of **BMDScannerTimeScale**.
timeInFrame	out	Time in frame in units of **BMDScannerTimeScale**.
ticksPerFrame	out	Number of ticks for a frame in units of **BMDScannerTimeScale**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.8 IScannerInputCallback Interface

The **IScannerInputCallback** object interface is a callback class which is called for each captured frame. An object with an **IScannerInputCallback** interface may be registered as a callback with the IscannerInput object interface.

Related Interfaces

Interface	Interface ID	Description
`IScannerInput`	`IID_IScanner`	New input frames are returned to IScannerInputCallback:: ScannerInputFrameArrived by the IScannerInput interface.
`IScannerInputFrame`	`IID_IScanner InputFrame`	IScannerInputFrame subclasses IScannerInputCallback.

Public Member Functions	
Method	**Description**
ScannerInputFrameArrived	Called when new video data is available.
ScannerInputFormatChanged	Called when a video input format change is detected.

2.8.1 IScannerInputCallback::ScannerInputFrameArrived method

The **ScannerInputFrameArrived** method is called when a video input frame 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    ScannerInputFrameArrived (IScannerInputFrame *videoFrame);
```

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 or assign it to a new COMPtr. The video frame can be released at any time by calling Release, or by having used a COMPtr and simply assigning it to nullptr.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.8.2 IScannerInputCallback::ScannerInputFormatChanged method

The **ScannerInputFormatChanged** method is called when a video input format change has been detected by the hardware. To enable this feature, the **bmdScannerVideoInputEnableFormatDetection** flag must set when calling **IScannerInput::EnableVideoInput()**.

Syntax

```
HRESULT    ScannerInputFormatChanged (BMDScannerDisplayMode newDisplayMode);
```

Parameters

Name	Direction	Description
`newDisplayMode`	in	The new display mode.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9 IScannerFilmConfiguration Interface

The **IScannerFilmConfiguration** object interface represents the current film configuration. Query it from the **IScanner** interface with **IID_IScannerFilmConfiguration** and then use the returned **IScannerFilmConfiguration** interface to access its methods described below.

Related Interfaces

Interface	Interface ID	Description
`IScanner`	`IID_IScanner`	Query the IID_IScannerFilmConfiguration interface from the IScanner interface. If a firmware update is required, this interface returns E_ACCESSDENIED when queried. Run the Cintel SScanner software utility to update the scanner firmware.

Public Member Functions	
Method	**Description**
SetFilmType	Configure the scanner for positive, negative, interpositive, or internegative film.
GetFilmType	Return the type of film the scanner is set to scan, such as positive.
GetDefaultFilmType	Returns the default type of film the scanner will scan.
GetFilmGaugeAvailable	Returns whether the specified film gauge is supported.
SetFilmGauge	Specify the type of film to scan, such 35mm 4 perf.
GetFilmGauge	Return the gauge of film the scanner is set to scan, such as 35mm 4 perf.
GetDefaultFilmGauge	Returns the default film gauge for the scanner.
SetFilmFrameRate	Set the frame rate of the film being scanned. Davinci Resolve uses this automatically to interpolate the final rendered output framerate and timecode accordingly.
GetFilmFrameRate	Return the frame rate of the film being scanned.

Public Member Functions	
Method	**Description**
GetDefaultFilmFrameRate	Returns the default frame rate that the film being scanned was shot at.
GetMinimumFilmFrameRate	Return the minimum possible film frame rate for the current film configuration. This minimum is enforced by the firmware.
GetMaximumFilmFrameRate	Return the maximum possible film frame rate for the current film configuration. This maximum is enforced by the firmware.
SetFeedWind	Set the wind direction of the supply spool A wind or B wind
GetFeedWind	Returns the wind direction of the feed spool.
GetDefaultFeedWind	Returns the default wind off direction for the feed reel.
SetTakeupWind	Set the wind direction of the takeup spool as A wind or B wind
GetTakeupWind	Returns the wind direction of the takeup spool.
GetDefaultTakeupWind	Returns the default wind on direction for the take up reel.
SetFeedRollType	Set the feed spool roll type to either core or reel. This is important, as different motor parameters are required for reels as they have significantly less mass.
GetFeedRollType	Return the feed spool roll type.
GetDefaultFeedRollType	Returns the default type of the film reel for the feed spool.
SetTakeupRollType	Set the takeup spool roll type to either core or reel. This is important, as different motor parameters are required for reels as they have significantly less mass.
GetTakeupRollType	Return the type of film reel for the takeup spool.
GetDefaultTakeupRollType	Returns the default type of film reel for the takeup spool.

2.9.1 IScannerFilmConfiguration::SetFilmType method

The **SetFilmType** method configures the scanner for positive, negative, interpositive, or internegative film.

Syntax

```
HRESULT    SetFilmType (BMDScannerFilmType filmType);
```

Parameters

Name	Direction	Description
`filmType`	in	**BMDScannerFilmType** specifying the film type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no film type configuration setting for this operation corresponding to the given **BMDScannerFilmType**.

2.9.2 IScannerFilmConfiguration::GetFilmType method

The **GetFilmType** method provides information about the type of film the scanner is set to scan.

Syntax

```
HRESULT     GetFilmType (BMDScannerFilmType *filmType);
```

Parameters

Name	Direction	Description
filmType	out	BMDScannerFilmType specifying the film type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.3 IScannerFilmConfiguration::GetDefaultFilmType method

The **GetDefaultFilmType** method returns the default type of film the scanner will scan.

Syntax

```
HRESULT     GetDefaultFilmType (BMDScannerFilmType, *filmType);
```

Parameters

Name	Direction	Description
filmType	out	BMDScannerFilmType specifying the default film type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.4 IScannerFilmConfiguration::GetFilmGaugeAvailable method

The **GetFilmGaugeAvailable** method returns whether the specified film gauge is supported.

Syntax

```
HRESULT GetFilmGaugeAvailable (BMDScannerFilmGauge filmGauge, bool *available);
```

Parameters

Name	Direction	Description
filmGauge	in	A BMDScannerFilmGauge film gauge to query.
available	out	True if the film gauge is available, false if unavailable.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.5 IScannerFilmConfiguration::SetFilmGauge method

The **SetFilmGauge** method lets you specify 35mm 2, 3, and 4 perf, or 16mm film.

Syntax

```
HRESULT     SetFilmGauge (BMDScannerFilmGauge filmGauge);
```

Parameters

Name	Direction	Description
filmGauge	in	BMDScannerFilmGauge specifying the film gauge.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no film gauge configuration setting for this operation corresponding to the given BMDScannerFilmGauge.

2.9.6 IScannerFilmConfiguration::GetFilmGauge method

The **GetFilmGauge** method returns the gauge of film the scanner is set to scan, such as 35mm 4 perf.

Syntax

```
HRESULT     GetFilmGauge (BMDScannerFilmGauge *filmGauge);
```

Parameters

Name	Direction	Description
filmGauge	out	BMDScannerFilmGauge specifying the film gauge.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.7 IScannerFilmConfiguration::GetDefaultFilmGauge method

The **GetDefaultFilmGauge** method returns the default film gauge the scanner is set to scan.

Syntax

```
HRESULT     GetDefaultFilmGauge (BMDScannerFilmGauge, *filmGauge);
```

Parameters

Name	Direction	Description
filmGauge	out	**BMDScannerFilmGauge** specifying the default film gauge.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.8 IScannerFilmConfiguration::SetFilmFrameRate method

The **SetFilmFrameRate** method sets the frame rate of the film being scanned. Resolve will use this automatically to interpolate the final rendered output framerate and timecode accordingly.

Syntax

```
HRESULT     SetFilmFrameRate (uint16_t framesPerSecond);
```

Parameters

Name	Direction	Description
framesPerSecond	in	The framerate of the film being scanned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.9 IScannerFilmConfiguration::GetFilmFrameRate method

The **GetFilmFrameRate** method returns the frame rate of the film being scanned.

Syntax

```
HRESULT     GetFilmFrameRate (uint16_t *framesPerSecond);
```

Parameters

Name	Direction	Description
framesPerSecond	out	The framerate of the film being scanned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.10 IScannerFilmConfiguration::GetDefaultFilmFrameRate method

The **GetDefaultFilmFrameRate** method returns the default frame rate that the film being scanned was shot at.

Syntax

```
HRESULT     GetDefaultFilmFrameRate (uint16_t, *framesPerSecond);
```

Parameters

Name	Direction	Description
framesPerSecond	out	The default frame rate of the film being scanned.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.11 IScannerFilmConfiguration::GetMinimumFilmFrameRate method

The **GetMinimumFilmFrameRate** method returns the minimum possible film frame rate for the current film configuration. This minimum is enforced by the firmware.

Syntax

```
HRESULT     GetMinimumFilmFrameRate (uint16_t *framesPerSecond);
```

Parameters

Name	Direction	Description
framesPerSecond	out	The minimum film frame rate of the current film configuration.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.12 IScannerFilmConfiguration::GetMaximumFilmFrameRate method

The **GetMaximumFilmFrameRate** method returns the maximum possible film frame rate for the current film configuration. This maximum is enforced by the firmware.

Syntax

```
HRESULT    GetMaximumFilmFrameRate (uint16_t *framesPerSecond);
```

Parameters

Name	Direction	Description
`framesPerSecond`	out	The maximum film frame rate of the current film configuration.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.13 IScannerFilmConfiguration::SetFeedWind method

The **SetFeedWind** method sets the wind direction of the supply spool on the left side of the scanner as anticlockwise or clockwise. Note that A wind unloads and takes up film from the left of the spool, while B wind unloads and takes up film from the right of the spool.

Syntax

```
HRESULT    SetFeedWind (BMDScannerWindType wind);
```

Parameters

Name	Direction	Description
`wind`	in	**BMDScannerWindType** specifying the feed wind type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no wind configuration setting for this operation corresponding to the given **BMDScannerWindType**.

2.9.14 IScannerFilmConfiguration::GetFeedWind method

The **GetFeedWind** method tells you whether the film is set to wind off the feed reel in a clockwise or anticlockwise direction.

Syntax

```
HRESULT     GetFeedWind (BMDScannerWindType *wind);
```

Parameters

Name	Direction	Description
wind	out	**BMDScannerWindType** specifying the feed wind type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.15 IScannerFilmConfiguration::GetDefaultFeedWind method

The **GetDefaultFeedWind** method returns the default wind off direction for the feed reel.

Syntax

```
HRESULT     GetDefaultFeedWind (BMDScannerWindType *wind);
```

Parameters

Name	Direction	Description
wind	out	**BMDScannerWindType** specifying the default feed reel wind direction.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.16 IScannerFilmConfiguration::SetTakeupWind method

The **SetTakeupWind** method sets the wind direction of the takeup spool on the right side of the scanner as anticlockwise or clockwise.

Syntax

```
HRESULT     SetTakeupWind (BMDScannerWindType wind);
```

Parameters

Name	Direction	Description
`wind`	in	**BMDScannerWindType** specifying the takeup wind type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no wind configuration setting for this operation corresponding to the given **BMDScannerWindType**.

2.9.17 IScannerFilmConfiguration::GetTakeupWind method

The **GetTakeupWind** method tells you whether the film is set to wind onto the takeup reel in a clockwise or anticlockwise direction.

Syntax

```
HRESULT     GetTakeupWind (BMDScannerWindType *wind);
```

Parameters

Name	Direction	Description
`wind`	out	**BMDScannerWindType** specifying the takeup wind type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.18 IScannerFilmConfiguration::GetDefaultTakeupWind method

The **GetDefaultTakeupWind** method returns the default wind on direction for the take up reel.

Syntax

```
HRESULT    GetDefaultTakeupWind (BMDScannerWindType *wind);
```

Parameters

Name	Direction	Description
`wind`	out	**BMDScannerWindType** specifying the default takeup reel wind direction.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.19 IScannerFilmConfiguration::SetFeedRollType method

The **SetFeedRollType** method sets the type of film reel for the feed spool to improve stability. Projection reels have a different weight and inertia compared to a core with a backplate, and this can affect the transport system.

Syntax

```
HRESULT    SetFeedRollType (BMDScannerRollType roll);
```

Parameters

Name	Direction	Description
`roll`	in	**BMDScannerRollType** specifying the feed roll type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no roll type configuration setting for this operation corresponding to the given **BMDScannerRollType**.

2.9.20 IScannerFilmConfiguration::GetFeedRollType method

The **GetFeedRollType** method returns the type of film reel for the feed spool.

Syntax

```
HRESULT    GetFeedRollType (BMDScannerRollType *roll);
```

Parameters

Name	Direction	Description
roll	out	BMDScannerRollType specifying the feed roll type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.21 IScannerFilmConfiguration::GetDefaultFeedRollType method

The **GetDefaultFeedRollType** method returns the default type of film reel for the feed spool.

Syntax

```
HRESULT    GetDefaultFeedRollType (BMDScannerRollType *roll);
```

Parameters

Name	Direction	Description
roll	out	BMDScannerRollType specifying the default feed spool roll type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.22 IScannerFilmConfiguration::SetTakeupRollType method

The **SetTakeupRollType** method sets the type of film reel for the takeup spool to improve stability. Projection reels have a different weight and inertia compared to a core with a backplate, and this can affect the transport system.

Syntax

```
HRESULT    SetTakeupRollType (BMDScannerRollType roll);
```

Parameters

Name	Direction	Description
`roll`	in	**BMDScannerRollType** specifying the takeup roll type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success
E_INVALIDARG	There is no roll type configuration setting for this operation corresponding to the given BMDScannerRollType.

2.9.23 IScannerFilmConfiguration::GetTakeupRollType method

The **GetTakeupRollType** method returns the type of film reel for the takeup spool.

Syntax

```
HRESULT    GetTakeupRollType (BMDScannerRollType *roll);
```

Parameters

Name	Direction	Description
`roll`	out	**BMDScannerRollType** specifying the takeup roll type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.9.24 IScannerFilmConfiguration::GetDefaultTakeupRollType method

The **GetDefaultTakeupRollType** method returns the default type of film reel for the takeup spool.

Syntax

```
HRESULT    GetDefaultTakeupRollType (BMDScannerRollType *roll);
```

Parameters

Name	Direction	Description
`roll`	out	**BMDScannerRollType** specifying the default takeup spool roll type.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10 IScannerTransport Interface

The **IScannerTransport** object interface represents the current transport configuration. Query it from the **IScanner** interface with **IID_IScannerTransport** and then use the returned **IScannerTransport** interface to access its methods described below.

Related Interfaces

Interface	Interface ID	Description
`IScanner`	`IID_Scanner`	Query the **IScannerTransport** interface from the **IScanner** interface. If a firmware update is required, this interface returns **E_ACCESSDENIED** when queried. Run the Cintel Scanner software to update the scanner firmware.

Public Member Functions	
Method	**Description**
SetRunSpeed	Set the scanning speed for the scanner.
GetRunSpeed	Return the current scanning speed of the scanner.
GetMinimumRunSpeed	Return the minimum scanning speed of the scanner.
GetMaximumRunSpeed	Return the maximum scanning speed of the scanner.
GetDefaultRunSpeed	Return the default scanning speed of the scanner.
SetShuttleSpeed	Set the speed of shuttling from one section of film to another.
GetShuttleSpeed	Returns the current shuttle speed of the scanner.
GetMinimumShuttleSpeed	Returns the minimum shuttle speed of the scanner.
GetMaximumShuttleSpeed	Returns the maximum shuttle speed of the scanner.
GetDefaultShuttleSpeed	Returns the default shuttle speed of the scanner.

Public Member Functions	
Method	**Description**
SetAcceleration	Set how quickly the scanning speed can change in frames per second per second.
GetAcceleration	Return the current setting for rate of change of shuttle speed in frames per second per second.
GetMinimumAcceleration	Return the minimum rate of change of shuttle speed in frames per second per second.
GetMaximumAcceleration	Return the maximum rate of change of shuttle speed in frames per second per second.
GetDefaultAcceleration	Return the default rate of change of shuttle speed in frames per second per second.
SetFilmTension	Set the film tension applied to 35mm film.
GetFilmTension	Return the film tension applied to 35mm film.
GetMinimumFilmTension	Return the minimum film tension applied to 35mm film.
GetMaximumFilmTension	Return the maximum film tension applied to 35mm film.
GetDefaultFilmTension	Return the default film tension applied to 35mm film.
SetTimecode	Set the timecode of the current frame of the roll of film you are about to scan.
GetTimecode	Return timecode metadata for the first frame of the roll of film.
SetCue	Set the timecode of a ‘cue’ to go to. The scanner will shuttle to this frame immediately.
SendCommand	Send the desired transport command, such as run, stop or frame up.
SetCaptureMode	Attempt to set the desired capture mode. The scanner defaults to ‘preview’ mode, which is not suitable for capturing to disk as the audio will not be frame accurate.
GetCaptureMode	Get the current capture mode.
GetDefaultCaptureMode	Return the default capture mode.
GetFeedFramesRemaining	Estimate of how many frames remain on the feed spool to the nearest core size.
GetTakeupFramesRemaining	Estimate of how many frames remain on the takeup spool to the nearest core size.
GetPreroll	Get the number of preroll frames required to make sure audio and video are captured properly when entering the desired capture mode.
SetAdvancedFlags	Set flags for advanced features of the scanner. See **BMDScannerAdvancedFlag** for more details.
GetAdvancedFlags	Return flags for advanced features of the scanner. See **BMDScannerAdvancedFlag** for more details.
GetHDRCaptureAvailable	Return whether HDR capture is currently available and the necessary steps were successful.

2.10.1 IScannerTransport::SetRunSpeed method

The **SetRunSpeed** method sets the scan speed. With adequate disk performance, you should be able to scan at 30 fps. However, if you're scanning to a slow hard drive, you can reduce the scanning speed to a frame rate that's suitable for your workstation without dropping frames.

Note: When scanning interpositive and internegative film, the increased density of the film requires slightly extended pulse durations from the light source. Normally, this does not affect the scan, however, a slight reduction in resolution may occur when scanning at above 12 frames per second. If you notice a difference in resolution, reduce your scanning speed to 12 frames per second or less.

Syntax

```
HRESULT     SetRunSpeed (uint16_t runSpeed);
```

Parameters

Name	Direction	Description
runSpeed	in	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.2 IScannerTransport::GetRunSpeed method

The **GetRunSpeed** method returns the current scanning speed of the scanner.

Syntax

```
HRESULT     GetRunSpeed (uint16_t *runSpeed);
```

Parameters

Name	Direction	Description
runSpeed	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.3 IScannerTransport::GetMinimumRunSpeed method

The **GetMinimumRunSpeed** method returns the setting for the minimum scanning speed of the scanner.

Syntax

```
HRESULT     GetMinimumRunSpeed (uint16_t *minimumRunSpeed);
```

Parameters

Name	Direction	Description
`minimumRunSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.4 IScannerTransport::GetMaximumRunSpeed method

The **GetMaximumRunSpeed** method returns the setting for the maximum scanning speed of the scanner.

Syntax

```
HRESULT     GetMaximumRunSpeed (uint16_t *maximumRunSpeed);
```

Parameters

Name	Direction	Description
`maximumRunSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.5 IScannerTransport::GetDefaultRunSpeed method

The **GetDefaultRunSpeed** method returns the setting for the default scanning speed of the scanner.

Syntax

```
HRESULT     GetDefaultRunSpeed (uint16_t *defaultRunSpeed);
```

Parameters

Name	Direction	Description
`defaultRunSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.6 IScannerTransport::SetShuttleSpeed method

The **SetShuttleSpeed** method sets the speed of shuttling from one section of film to another. Recommended speeds are 1-100 frames per second for 35mm film, and 1-200 frames per second for 16mm film.

Note: Fast acceleration and shuttle speeds can be hard on archival footage, so it's recommended to lower the maximum acceleration and maximum speed whenever you're scanning older film.

Syntax

```
HRESULT     SetShuttleSpeed (uint16_t shuttleSpeed);
```

Parameters

Name	Direction	Description
`shuttleSpeed`	in	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.7 IScannerTransport::GetShuttleSpeed method

The **GetShuttleSpeed** method returns the current setting for the shuttle speed of the scanner.

Syntax

```
HRESULT     GetShuttleSpeed (uint16_t *shuttleSpeed);
```

Parameters

Name	Direction	Description
`shuttleSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.8 IScannerTransport::GetMinimumShuttleSpeed method

The **GetMinimumShuttleSpeed** method returns the minimum of the range of settings for the shuttle speed of the scanner.

Syntax

```
HRESULT    GetMinimumShuttleSpeed (uint16_t *minimumShuttleSpeed);
```

Parameters

Name	Direction	Description
`minimumShuttleSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.9 IScannerTransport::GetMaximumShuttleSpeed method

The **GetMaximumShuttleSpeed** method returns the maximum of the range of settings for the shuttle speed of the scanner.

Syntax

```
HRESULT    GetMaximumShuttleSpeed (uint16_t *maximumShuttleSpeed);
```

Parameters

Name	Direction	Description
`maximumShuttleSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.10 IScannerTransport::GetDefaultShuttleSpeed method

The **GetDefaultShuttleSpeed** method returns the default shuttle speed of the scanner.

Syntax

```
HRESULT    GetDefaultShuttleSpeed (uint16_t *defaultShuttleSpeed);
```

Parameters

Name	Direction	Description
`defaultShuttleSpeed`	out	Frames per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.11 IScannerTransport::SetAcceleration method

The **SetAcceleration** method sets how quickly the scanning speed can change in frames per second per second.

> **NOTE** Fast acceleration and shuttle speeds can be harsh on archival footage, so it's recommended to lower the maximum acceleration and maximum speed whenever you're scanning older film.

Syntax

```
HRESULT     SetAcceleration (uint16_t acceleration);
```

Parameters

Name	Direction	Description
`acceleration`	in	Rate of change in frames per second per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.12 IScannerTransport::GetAcceleration method

The **GetAcceleration** method returns the current rate of change of shuttle speed in frames per second per second.

Syntax

```
HRESULT     GetAcceleration (uint16_t *acceleration);
```

Parameters

Name	Direction	Description
`acceleration`	out	Rate of change of shuttle speed in frames per second per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.13 IScannerTransport::GetMinimumAcceleration method

The **GetMinimumAcceleration** method returns the maximum rate of change of shuttle speed in frames per second per second.

Syntax

```
HRESULT    GetMinimumAcceleration (uint16_t *minimumAcceleration);
```

Parameters

Name	Direction	Description
`minimumAcceleration`	out	Rate of change of shuttle speed in frames per second per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.14 IScannerTransport::GetMaximumAcceleration method

The **GetMaximumAcceleration** method returns the maximum rate of change of shuttle speed in frames per second per second.

Syntax

```
HRESULT    GetMaximumAcceleration (uint16_t *maximumAcceleration);
```

Parameters

Name	Direction	Description
`maximumAcceleration`	out	Rate of change of shuttle speed in frames per second per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.15 IScannerTransport::GetDefaultAcceleration method

The **GetDefaultAcceleration** method returns the rate of change of shuttle speed in frames per second per second.

Syntax

```
HRESULT    GetDefaultAcceleration (uint16_t *defaultAcceleration);
```

Parameters

Name	Direction	Description
`defaultAcceleration`	out	Rate of change of shuttle speed in frames per second per second

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.16 IScannerTransport::SetFilmTension method

The **SetFilmTension** method makes very small changes to gently adjust the film tension to prevent sprocket picking and applied to 35mm film. Note that this parameter is not supported for 16mm film. The normalized value is an adjustment value over the full scale of the provided type and does not relate to a physical measure such as grams.

Syntax

```
HRESULT    SetFilmTension (uint16_t tension);
```

Parameters

Name	Direction	Description
`tension`	in	Normalized value

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.17 IScannerTransport::GetFilmTension method

The **GetFilmTension** method returns the amount of tension applied to 35mm film. Note that this parameter is not supported for 16mm film. The normalized value is an adjustment value over the full scale of the provided type and does not relate to a physical measure such as grams.

Syntax

```
HRESULT    GetFilmTension (uint16_t *tension);
```

Parameters

Name	Direction	Description
`tension`	out	Normalized value

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.18 IScannerTransport::GetMinimumFilmTension method

The **GetMinimumFilmTension** method returns the minimum value for the range of tension that can be applied to 35mm film. Note that this parameter is not supported for 16mm film. The normalized value is an adjustment value over the full scale of the provided type and does not relate to a physical measure such as grams.

Syntax

```
HRESULT    GetMinimumFilmTension (uint16_t *minimumTension);
```

Parameters

Name	Direction	Description
`minimumTension`	out	Normalized value

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.19 IScannerTransport::GetMaximumFilmTension method

The **GetMaximumFilmTension** method returns the maximum value in the range of tension that can be applied to 35mm film. Note that this parameter is not supported for 16mm film. The normalized value is an adjustment value over the full scale of the provided type and does not relate to a physical measure such as grams.

Syntax

```
HRESULT    GetMaximumFilmTension (uint16_t *maximumTension);
```

Parameters

Name	Direction	Description
`maximumTension`	out	Normalized value

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.20 IScannerTransport::GetDefaultFilmTension method

The **GetDefaultFilmTension** method returns the default value for the tension that can be applied to 35mm film. Note that this parameter is not supported for 16mm film. The normalized value is an adjustment value over the full scale of the provided type and does not relate to a physical measure such as grams.

Syntax

```
HRESULT    GetDefaultFilmTension (uint16_t *defaultTension);
```

Parameters

Name	Direction	Description
`defaultTension`	out	Normalized value

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.21 IScannerTransport::SetTimecode method

The **SetTimecode** method sets the timecode of the current frame of the roll of film you are about to scan. For example, if you're scanning the first roll of a project, you can enter 01:00:00:00.

Before the first frame, the zero frame for the roll usually has a small hole punched in the frame as a permanent reference. This is referred to as the marker frame, lab roll hole, or head punch. By always setting the first frame of timecode to match the marker frame, subsequent film scans have the same frame count as previous scans, making it possible to rescan and reconform the same material whenever necessary.

Syntax

```
HRESULT     SetTimecode (BMDScannerTimecodeBCD timecodeBcd);
```

Parameters

Name	Direction	Description
`timecodeBcd`	in	**BMDScannerTimecodeBCD** to apply to the current frame.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.22 IScannerTransport::GetTimecode method

The **GetTimecode** method returns the timecode metadata for the first frame of the roll of film.

Syntax

```
HRESULT     GetTimecode (BMDScannerTimecodeBCD *timecodeBcd);
```

Parameters

Name	Direction	Description
`timecodeBcd`	out	**BMDScannerTimecodeBCD** of the current frame.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.23 IScannerTransport::SetCue method

The **SetCue** method sets the timecode of a 'cue' to go to. The scanner will shuttle to this frame immediately.

Syntax

```
HRESULT     SetCue (BMDScannerTimecodeBCD timecodeBcd);
```

Parameters

Name	Direction	Description
timecodeBcd	in	BMDScannerTimecodeBCD to shuttle to.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.24 IScannerTransport::SendCommand method

The **SendCommand** method sends a transport command you specify, such as run, stop, frame up, and so on.

Syntax

```
HRESULT     SendCommand (BMDScannerTransportCommand command);
```

Parameters

Name	Direction	Description
command	in	BMDScannerTransportCommand to execute.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.25 IScannerTransport::SetCaptureMode method

The **SetCaptureMode** method instructs the scanner to enter the specified capture mode. This may not happen immediately. Here is a summary of capture modes defined in **BMDScannerCaptureMode**:

Preview: useful for live preview but unsuitable for capture.

Normal Exposure: standard capture with synchronized audio.

High Exposure: high exposure pass for HDR.

After you use **SetCaptureMode**, it is a good practice to check the **SystemStateChanged** notifier to confirm successful entry into the capturing state, or to see information about why it may be failing.

Syntax

```
HRESULT     SetCaptureMode (BMDScannerCaptureMode mode);
```

Parameters

Name	Direction	Description
mode	in	BMDScannerCaptureMode to attempt to use when capturing frames.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.26 IScannerTransport::GetCaptureMode method

The **GetCaptureMode** method returns a capture mode that was set by **SetCaptureMode**. If it is not set, the default is 'preview' mode. See **BMDScannerCaptureMode**.

Syntax

```
HRESULT     GetCaptureMode (BMDScannerCaptureMode *mode);
```

Parameters

Name	Direction	Description
mode	out	BMDScannerCaptureMode currently active.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.27 IScannerTransport::GetDefaultCaptureMode method

The **GetDefaultCaptureMode** method returns the default scanner capture mode. If the capture mode has not been specified by **SetCaptureMode**, the mode should be 'preview'.

Syntax

```
HRESULT    GetDefaultCaptureMode (BMDScannerCaptureMode *mode);
```

Parameters

Name	Direction	Description
mode	out	Default scanner capture mode.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.28 IScannerTransport::GetFeedFramesRemaining method

The **GetFeedFramesRemaining** method returns an estimate of how many frames are remaining on the feed spool.

Syntax

```
HRESULT    GetFeedFramesRemaining (uint32_t *frames);
```

Parameters

Name	Direction	Description
frames	out	Estimated number of frames.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.29 IScannerTransport::GetTakeupFramesRemaining method

The **GetTakeupFramesRemaining** method returns an estimate of how many frames are remaining on the takeup spool.

Syntax

```
HRESULT    GetTakeupFramesRemaining (uint32_t *frames);
```

Parameters

Name	Direction	Description
`frames`	out	Estimated number of frames.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.30 IScannerTransport::GetPreroll method

The **GetPreroll** method returns the minimum number of preroll frames necessary to confirm audio and video will be captured properly in the current capture mode.

Syntax

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

Parameters

Name	Direction	Description
`frames`	out	Number of frames.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.31 IScannerTransport::SetAdvancedFlags method

The **SetAdvancedFlags** method sets bitmask flags to enable or disable advanced features of the scanner. See **BMDScannerAdvancedFlag**.

Syntax

```
HRESULT    SetAdvancedFlags (uint32_t flags);
```

Parameters

Name	Direction	Description
flags	in	Set BMDScannerAdvancedFlag binary masking.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.32 IScannerTransport::GetAdvancedFlags method

The **GetAdvancedFlags** method returns bitmask flags to enable or disable advanced features of the scanner. See **BMDScannerAdvancedFlag**.

Syntax

```
HRESULT    GetAdvancedFlags (uint32_t *flags);
```

Parameters

Name	Direction	Description
flags	out	BMDScannerAdvancedFlag currently set in binary masking.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.10.33 IScannerTransport::GetHDRCaptureAvailable method

The **GetHDRCaptureAvailable** method returns whether HDR capture is currently available. For example, HDR requires that light calibration has been run successfully, so this method tests for those conditions.

Syntax

```
HRESULT    GetHDRCaptureAvailable (bool *available);
```

Parameters

Name	Direction	Description
`available`	out	True if HDR is available, false if not. Observe the notifiers to understand why it may not be available.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11 IScannerColorProperties Interface

The **IScannerColorProperties** object interface represents the current color properties configuration. Query it from the **IScanner** interface with **IID_IScannerColorProperties** and then use the returned **IScannerColorProperties** interface to access its methods described below.

Related Interfaces

Name	Direction	Description
`IScanner`	`IID_IScanner`	Query the **IID_IScannerColorProperties** interface from the **IScanner** interface. If a firmware update is required, this interface returns **E_ACCESSDENIED** when queried. Run the Cintel Scanner software to update the scanner firmware.

Public Member Functions	
Method	**Description**
SetLEDIllumination	Set RGB levels for the LED light source.
GetLEDIllumination	Return RGB levels for calibration of the LED light source.
GetDefaultLEDIllumination	Return the default RGB values for the LED light source.
StartFixedPatternGainCalibration	With the scanner unloaded and the gate free of film, run the per pixel gain calibration to remove any blemishes from the image path. Observe notifiers for failure.
GetFixedPatternGainCalibrated	Return whether the scanner has fixed pattern gain calibration.
StartLEDCalibration	Begin ‘auto black’ or ‘auto white’ automatic calibration of the LED light source.
GetLEDCalibrationStatus	Return the status of ‘auto black’ or ‘auto white’ automatic calibration of the LED light source.

Public Member Functions	
Method	**Description**
GetHDRCalibrationStatus	Return the status of the ‘auto black’ or ‘auto white’ automatic calibration of the LED light source for the high exposure HDR image. This is done automatically if HDR is enabled.
GetFixedPatternGainCalibrationStatus	Return the status of the fixed pattern gain calibration.
SetFocusPeakingEnable	Enable the ‘focus assist’ feature and show when the film plane is in focus. The luma peaking algorithm applies to the HDMI output only. You can make a feature similar to that found in Davinci Resolve if required for live preview on images received via this SDK.
GetFocusPeakingEnable	Return whether the ‘focus assist’ feature is enabled.
GetDefaultFocusPeakingEnable	Return whether the ‘focus assist’ feature is enabled by default.
SetEnableHDRMode	Enable HDR dual pass scanning.
GetEnableHDRMode	Return whether HDR dual pass scanning is enabled.
GetDefaultEnableHDRMode	Return whether HDR dual pass scanning is enabled by default.

2.11.1 IScannerColorProperties::SetLEDIllumination method

The **SetLEDIllumination** method adjusts the scanner’s light source to set intensity and colour temperature.

Syntax

```
HRESULT    SetLEDIllumination (uint16_t red, uint16_t green, uint16_t blue);
```

Parameters

Name	Direction	Description
`red`	in	A scale of 0 to uint16_t maximum representing minimum to maximum intensity of the red LEDs.
`green`	in	A scale of 0 to uint16_t maximum representing minimum to maximum intensity of the green LEDs.
`blue`	in	A scale of 0 to uint16_t maximum representing minimum to maximum intensity of the blue LEDs.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.2 IScannerColorProperties::GetLEDIllumination method

The **GetLEDIllumination** method returns RGB values for the LED light source.

Syntax

```
HRESULT    GetLEDIllumination (uint16_t *red, uint16_t *green, uint16_t *blue);
```

Parameters

Name	Direction	Description
red	out	A scale of 0 to uint16_t maximum representing minimum to maximum intensity of the red LEDs.
green	out	A scale of 0 to uint16_t maximum representing minimum to maximum intensity of the green LEDs.
blue	out	A scale of 0 to uint16_t maximum representing minimum to maximum intensity of the blue LEDs.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.3 IScannerColorProperties::GetDefaultLEDIllumination method

The **GetDefaultLEDIllumination** method returns the default RGB values for the LED light source.

Syntax

```
HRESULT    GetDefaultLEDIllumination (uint16_t *red, uint16_t *green,
                                      uint16_t *blue);
```

Parameters

Name	Direction	Description
red	out	A uint16_t representing the default intensity of the red LEDs.
green	out	A uint16_t representing the default intensity of the green LEDs.
blue	out	A uint16_t representing the default intensity of the blue LEDs.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.4 IScannerColorProperties::StartFixedPatternGainCalibration method

The **StartFixedPatternGainCalibration** method runs the per pixel gain calibration to remove any blemishes from the image path.

Syntax

```
HRESULT    StartFixedPatternGainCalibration (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.5 IScannerColorProperties::GetFixedPatternGainCalibrated method

The **GetFixedPatternGainCalibrated** method returns whether the scanner has had fixed pattern gain calibration.

Syntax

```
HRESULT    GetFixedPatternGainCalibrated (bool *calibrated);
```

Parameters

Name	Direction	Description
`calibrated`	out	True if the scanner has been calibrated since being powered on, false otherwise. Observe notifiers for reasons of failure.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.6 IScannerColorProperties::StartLEDCalibration method

The **StartLEDCalibration** method analyzes the current frame and does an automatic adjustment to set the black point for negative film or set the white point for a film print.

On completion, it returns success or failure, or you can query the **GetLEDCalibrationStatus** method at any time.

If you don't use automatic calibration, use the **SetLEDIllumination** method to set the RGB values for the color and intensity manually.

Syntax

```
HRESULT    StartLEDCalibration (void);
```

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.7 IScannerColorProperties::GetLEDCalibrationStatus method

The **GetLEDCalibrationStatus** method returns the status of auto calibration.

Syntax

```
HRESULT    GetLEDCalibrationStatus (BMDScannerCalibrationStatus *status);
```

Parameters

Name	Direction	Description
`status`	out	BMDScannerCalibrationStatus of the current status. Observe notifiers to help understand reasons for failure.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.8 IScannerColorProperties::GetHDRCalibrationStatus method

The **GetHDRCalibrationStatus** method returns the status of HDR calibration. This is done after normal LED calibration if HDR is enabled.

Whenever you enable HDR, the auto black/white calibration is reset. This additional calibration of the LED light source helps to achieve high quality scans and inform you of any problems before the start of HDR scanning.

Syntax

```
HRESULT    GetHDRCalibrationStatus (BMDScannerCalibrationStatus *status);
```

Parameters

Name	Direction	Description
`status`	out	BMDScannerCalibrationStatus of the current status. Observe notifiers to help understand reasons for failure.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.9 IScannerColorProperties::GetFixedPatternGainCalibrationStatus method

The **GetFixedPatternGainCalibrationStatus** method returns the status of the per pixel gain calibration. See **BMDScannerCalibrationStatus**.

Syntax

```
HRESULT     GetFixedPatternGainCalibrationStatus
            (BMDScannerCalibrationStatus *status);
```

Parameters

Name	Direction	Description
`status`	out	**BMDScannerCalibrationStatus** of the current status. Observe notifiers to help understand reasons for failure.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.10 IScannerColorProperties::SetFocusPeakingEnable method

The **SetFocusPeakingEnable** method superimposes a focus peaking overlay over the Ultra HD image sent from the scanner's HDMI output. Focus peaking detects film grain of the scanned image whenever the film plane is in perfect focus, which helps the operator to focus the scanner even if the film image is out of focus.

The overlay generated by this luma based algorithm is superimposed on the HDMI output for your preview monitor. The focus peaking overlay does not affect scanned images in your saved clips.

Davinci Resolve applies this focus peaking overlay during live preview. If you are adding a similar feature in your own application with this SDK and want to know more about the filters used by Blackmagic Design, contact us on the Blackmagic Software Developers Forum.

Syntax

```
HRESULT     SetFocusPeakingEnable (bool enable);
```

Parameters

Name	Direction	Description
`enable`	in	Set to true to enable, or false to disable it.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.11 IScannerColorProperties::GetFocusPeakingEnable method

The **GetFocusPeakingEnable** method returns whether the 'focus assist' feature is enabled.

Syntax

```
HRESULT    GetFocusPeakingEnable (bool *enable);
```

Parameters

Name	Direction	Description
enable	out	True if enabled, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.12 IScannerColorProperties::GetDefaultFocusPeakingEnable method

The **GetDefaultFocusPeakingEnable** method returns the default setting for whether the 'focus assist' feature is enabled.

Syntax

```
HRESULT    GetDefaultFocusPeakingEnable (bool, *enable);
```

Parameters

Name	Direction	Description
enable	out	True if enabled, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.13 IScannerColorProperties::SetEnableHDRMode method

The **SetEnableHDRMode** method sets the scanner into HDR mode and checks for prerequisites such as LED calibration.

Syntax

```
HRESULT    SetEnableHDRMode (bool enable);
```

Parameters

Name	Direction	Description
enable	in	Set to true to enable HDR dual pass scanning, or false to remain in normal scanning mode.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.14 IScannerColorProperties::GetEnableHDRMode method

The **GetEnableHDRMode** method returns a boolean value corresponding to the status of the scanner's HDR mode.

Syntax

```
HRESULT    GetEnableHDRMode (bool *enable);
```

Parameters

Name	Direction	Description
`enable`	out	True if HDR dual pass scanning is enabled, or false for single pass scanning with standard dynamic range.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.11.15 IScannerColorProperties::GetDefaultEnableHDRMode method

The **GetDefaultEnableHDRMode** method returns whether HDR dual pass scanning mode is enabled by default.

Syntax

```
HRESULT    GetDefaultEnableHDRMode (bool, *enable);
```

Parameters

Name	Direction	Description
`enable`	out	True if HDR dual pass scanning is enabled by default, or false if single pass scanning with standard dynamic range is the default.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12 IScannerImageStabilizer Interface

The **IScannerImageStabilizer** object interface represents the current image stabilizer configuration. Query it from the **IScanner** interface with **IID_IScannerImageStabilizer** and then use the returned **IScannerImageStabilizer** interface to access its methods described below.

Related Interfaces

Interface	Interface ID	Description
`IScanner`	`IID_IScanner`	Query the **IID_IScannerImageStabilizer** interface from the **IScanner** interface. If a firmware update is required, this interface returns **E_ACCESSDENIED** when queried. Run the Cintel Scanner software to update the scanner firmware.

Public Member Functions	
Method	**Description**
SetStabilizationEnable	Enable the scanner's firmware based image stabilization. Note that the firmware does not alter the image. This setting enables calculations that are required by image stabilization in post processing.
GetStabilizationEnable	Returns whether the scanner's firmware based image stabilization is enabled.
GetDefaultStabilizationEnable	Returns the default setting for enabling or disabling the image stabilization system.
SetStabilizationEnableXY	Independently enable hardware image stabilization to fix horizontal gate weave and vertical gate hop respectively.
GetStabilizationEnableXY	Returns whether the scanner's hardware-based image stabilization is enabled horizontally, or vertically, or both.
GetDefaultStabilizationEnableXY	Returns the default setting for enabling or disabling image stabilization on the X axis, Y axis, or both.
SetStabilizationViewRegion	Enable the stabilizer's overlay.
GetStabilizationViewRegion	Return whether the stabilizer's overlay is visible.
GetDefaultStabilizationViewRegion	Return the default setting of the stabilizer's overlay view.
SetStabilizationOffsetX	Set a manual offset for the horizontal stabilizer.
GetStabilizationOffsetX	Return the amount of manual offset applied to the horizontal stabilizer.
GetMinimumStabilizationOffsetX	Return the minimum amount of manual offset for the horizontal stabilizer.
GetMaximumStabilizationOffsetX	Return the maximum amount of manual offset for the horizontal stabilizer.
GetDefaultStabilizationOffsetX	Return the default amount of manual offset applied to the horizontal stabilizer. It's important to note that this default amount changes based on film type and gate type.

2.12.1 IScannerImageStabilizer::SetStabilizationEnable method

The **SetStabilizationEnable** method enables hardware stabilization to eliminate horizontal gate weave and vertical film hop.

When image stabilization is enabled, a detection overlay is displayed in the viewer to highlight the edge of the film perforation that is used as the reference for stabilization. If the perforations are in a poor condition, you may want to disable hardware stabilization.

Syntax

```
HRESULT    SetStabilizationEnable (bool enable);
```

Parameters

Name	Direction	Description
enable	in	Set to true to enable the firmware based image stabilizer, or false to disable it.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.2 IScannerImageStabilizer::GetStabilizationEnable method

The **GetStabilizationEnable** method returns whether hardware stabilization system is enabled.

Syntax

```
HRESULT    GetStabilizationEnable (bool *enable);
```

Parameters

Name	Direction	Description
enable	out	Returns true if the firmware based image stabilizer is enabled, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.3 IScannerImageStabilizer::GetDefaultStabilizationEnable method

The **GetDefaultStabilizationEnable** method returns the default setting for enabling or disabling the image stabilization system.

Syntax

```
HRESULT    GetDefaultStabilizationEnable (bool *enable);
```

Parameters

Name	Direction	Description
`enable`	out	Return the default setting for the stabilizers.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.4 IScannerImageStabilizer::GetDefaultStabilizationEnableXY method

The **GetDefaultStabilizationEnableXY** method individually enables or disables the X and Y axes of the hardware image stabilization system to fix horizontal gate weave, or vertical gate hop, or both.

Note that the stabilizer offset setting is important for good horizontal weave behavior. The default setting is sufficient for general purpose performance.

Syntax

```
HRESULT    GetDefaultStabilizationEnableXY (bool enableX, bool enableY);
```

Parameters

Name	Direction	Description
`enableX`	in	Set to true to enable horizontal gate weave firmware based estimation, false otherwise.
`enableY`	in	Set to true to enable vertical gate hop firmware based estimation, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.5 IScannerImageStabilizer::GetStabilizationEnableXY method

The **GetStabilizationEnableXY** method returns the states of the X and Y axes of the hardware image stabilization system.

Syntax

```
HRESULT    GetStabilizationEnableXY (bool *enableX, bool *enableY);
```

Parameters

Name	Direction	Description
`enableX`	out	Return whether firmware based estimation is enabled for horizontal gate weave.
`enableY`	out	Return whether firmware based estimation is enabled for vertical gate hop.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.6 IScannerImageStabilizer::GetDefaultStabilizationEnableXY method

The **GetDefaultStabilizationEnableXY** method returns the default states of the X and Y axes of the hardware image stabilization system.

Syntax

```
HRESULT    GetDefaultStabilizationEnableXY (bool *enableX, bool *enableY);
```

Parameters

Name	Direction	Description
`enableX`	out	Return the default state for whether firmware based estimation is enabled for horizontal gate weave.
`enableY`	out	Return the default state for whether firmware based estimation is enabled for vertical gate hop.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.7 IScannerImageStabilizer::SetStabilizationViewRegion method

The **SetStabilizationViewRegion** method makes the stabilizer overlay visible.

Syntax

```
HRESULT    SetStabilizationViewRegion (bool enable);
```

Parameters

Name	Direction	Description
enable	in	Set to true to display the stabilizer overlay.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.8 IScannerImageStabilizer::GetStabilizationViewRegion method

The **GetStabilizationViewRegion** method returns whether the stabilizer overlay is displayed.

Syntax

```
HRESULT    GetStabilizationViewRegion (bool *enable);
```

Parameters

Name	Direction	Description
enable	out	True if stabilizer overlay is displayed.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.9 IScannerImageStabilizer::GetDefaultStabilizationViewRegion method

The **GetDefaultStabilizationViewRegion** method returns returns the default setting of the stabilizer's overlay view.

Syntax

```
HRESULT    GetDefaultStabilizationViewRegion (bool *enable);
```

Parameters

Name	Direction	Description
enable	out	True if stabilizer overlay is enabled by default.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.10 IScannerImageStabilizer::SetStabilizationOffsetX method

The **SetStabilizationOffsetX** method adjusts the horizontal position of the stabilization overlay. This lets you precisely adjust the alignment overlay to sit on the edge of the perforations.

Syntax

```
HRESULT     SetStabilizationOffsetX (int16_t stabilizationOffsetX);
```

Parameters

Name	Direction	Description
`stabilizationOffsetX`	in	Set the pixel offset between the edge of the image and the default for horizontal weave detection.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.11 IScannerImageStabilizer::GetStabilizationOffsetX method

The **GetStabilizationOffsetX** method returns the horizontal position of the stabilization overlay.

Syntax

```
HRESULT     GetStabilizationOffsetX (int16_t *stabilizationOffsetX);
```

Parameters

Name	Direction	Description
`stabilizationOffsetX`	out	Return the pixel offset from the default used for horizontal weave detection.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.12 IScannerImageStabilizer::GetMinimumStabilizationOffsetX method

The **GetMinimumStabilizationOffsetX** method returns the minimum horizontal offset of the stabilization overlay.

Syntax

```
HRESULT GetMinimumStabilizationOffsetX (int16_t *stabilizationOffsetX);
```

Parameters

Name	Direction	Description
`stabilizationOffsetX`	out	Return the minimum pixel offset from the default used for horizontal weave detection.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.13 IScannerImageStabilizer::GetMaximumStabilizationOffsetX method

The **GetMaximumStabilizationOffsetX** method returns the maximum horizontal offset of the stabilization overlay.

Syntax

```
HRESULT GetMaximumStabilizationOffsetX (int16_t *stabilizationOffsetX);
```

Parameters

Name	Direction	Description
`stabilizationOffsetX`	out	Return the maximum pixel offset from the default used for horizontal weave detection.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.12.14 IScannerImageStabilizer::GetDefaultStabilizationOffsetX method

The **GetDefaultStabilizationOffsetX** method returns the default amount of manual offset applied to the horizontal stabilizer.

Syntax

```
HRESULT    GetDefaultStabilizationOffsetX (int16_t *stabilizationOffsetX);
```

Parameters

Name	Direction	Description
`stabilizationOffsetX`	out	Default amount of manual offset applied to the horizontal stabilizer.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.13 IScannerCapture Interface

The **IScannerCapture** object interface represents the current capture configuration. Query it from the **IScanner** interface with **IID_IScannerCapture** and then use the returned **IScannerCapture** interface to access its methods described below.

Related Interfaces

Interface	Interface ID	Description
`IScanner`	`IID_IScanner`	Query the IID_IScannerCaptureInterface interface from the IScanner interface. If a firmware update is required, this interface returns E_ACCESSDENIED when queried. Run the Cintel Scanner software utility to update the scanner firmware.

Public Member Functions	
Method	**Description**
SetCompressionType	Set the type of codec to use when capturing footage.
GetCompressionType	Return the type of codec selected for capturing footage.
GetDefaultCompressionType	Return the default type of codec for capturing footage.

2.13.1 IScannerCapture::SetCompressionType method

The **SetCompressionType** method sets the type of codec to use when capturing footage. This is 'Cintel Raw' for lossless compression by default, or you can choose 'Cintel Raw 3:1' for even smaller file sizes.

Syntax

```
HRESULT    SetCompressionType (BMDScannerCompressionType type);
```

Parameters

Name	Direction	Description
`type`	in	BMDScannerCompressionType to set, for example LossLess, Lossy 3:1 or None.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.13.2 IScannerCapture::GetCompressionType method

The **GetCompressionType** method returns the codec that's been selected for capturing footage.

Syntax

```
HRESULT     GetCompressionType (BMDScannerCompressionType *type);
```

Parameters

Name	Direction	Description
type	out	Return the current **BMDScannerCompressionType**, for example LossLess, Lossy 3:1 or None.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.13.3 IScannerCapture::GetDefaultCompressionType method

The **GetDefaultCompressionType** method returns the default type of codec for capturing footage.

Syntax

```
HRESULT     GetDefaultCompressionType (BMDScannerCompressionType *type);
```

Parameters

Name	Direction	Description
type	out	**BMDScannerCompressionType** specifying the default acquisition codec for the scanner.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14 IScannerAudioSelect Interface

The **IScannerAudioSelect** object interface represents the external audio.

Related Interface

Interface	Interface ID	Description
`IScanner`	`IID_IScanner`	Query the IID_IscannerAudioSelect interface from the IScanner interface. If a firmware update is required, this interface returns E_ACCESSDENIED when queried. Run the Cintel Scanner software to update the scanner firmware.

Public Member Functions	
Method	**Description**
SetAudioSelectSource	Set the audio source.
GetAudioSelectSource	Return the current setting for the audio source.
GetDefaultAudioSelectSource	Return the default setting for the audio source.
GetAudioSourceAvailable	Return whether the specified audio source is available.
SetExternalAudioFormat	Set the external audio source format to analog or AES.
GetExternalAudioFormat	Return whether the external audio source format is analog or AES.
GetDefaultExternalAudioFormat	Return whether the default external audio source format is analog or AES.
SetSyncConfiguration	Set the current configuration of the Bi-phase/Timecode output.
GetSyncConfiguration	Return the current configuration of the Bi-phase/Timecode output.
GetDefaultSyncConfiguration	Return the default configuration of the Bi-phase/Timecode output.

2.14.1 IScannerAudioSelect::SetAudioSelectSource method

The **SetAudioSelectSource** method sets the audio source.

Syntax

```
HRESULT    SetAudioSelectSource (BMDScannerAudioSelectSource audioSelectSource);
```

Parameters

Name	Direction	Description
`audioSelectSource`	in	BMDScannerAudioSelectSource can be 'ASNo' for none, 'ASRe' for Blackmagic KeyKode and Audio Reader, or 'ASEx' for an external source.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.2 IScannerAudioSelect::GetAudioSelectSource method

The **GetAudioSelectSource** method returns the selected audio source for external audio.

Syntax

```
HRESULT GetAudioSelectSource (BMDScannerAudioSelectSource *audioSelectSource);
```

Parameters

Name	Direction	Description
audioSelectSource	out	BMDScannerAudioSelectSource can be 'ASNo' for none, 'ASRe' for Blackmagic KeyKode and Audio Reader, or 'ASEx' for an external source.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.3 IScannerAudioSelect::GetDefaultAudioSelectSource method

The **GetDefaultAudioSelectSource** method returns the default audio source for external audio.

Syntax

```
HRESULT     GetDefaultAudioSelectSource (BMDScannerAudioSelectSource
            *audioSelectSource);
```

Parameters

Name	Direction	Description
audioSelectSource	out	BMDScannerAudioSelectSource can be 'ASNo' for none, 'ASRe' for Blackmagic KeyKode and Audio Reader, or 'ASEx' for an external source.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.4 IScannerAudioSelect::GetAudioSourceAvailable method

The **GetAudioSourceAvailable** method returns whether the given **BMDScannerAudioSelectSource** external audio source is available.

Syntax

```
HRESULT     GetAudioSourceAvailable (BMDScannerAudioSelectSource
            audioSelectSource, bool *available);
```

Parameters

Name	Direction	Description
`audioSelectSource`	in	**BMDScannerAudioSelectSource** can be 'ASNo' for none, 'ASRe' for Blackmagic KeyKode and Audio Reader, or 'ASEx' for an external source.
`available`	out	True if available, false if unavailable.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.5 IScannerAudioSelect::SetExternalAudioFormat method

The **SetExternalAudioFormat** method sets the audio format for external audio recording.

Syntax

```
HRESULT     SetExternalAudioFormat (BMDScannerExternalAudioFormat
            externalAudioFormat);
```

Parameters

Name	Direction	Description
`externalAudioFormat`	in	**BMDScannerExternalAudioFormat** can be ‘EAAn’ for external analog or ‘EAAE’ for AES.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.6 IScannerAudioSelect::GetExternalAudioFormat method

The **GetExternalAudioFormat** method returns the audio format for external audio recording.

Syntax

```
HRESULT     GetExternalAudioFormat (BMDScannerExternalAudioFormat
            *externalAudioFormat);
```

Parameters

Name	Direction	Description
`externalAudioFormat`	out	BMDScannerExternalAudioFormat can be 'EAAn' for external analog or 'EAAE' for AES.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.7 IScannerAudioSelect::GetDefaultExternalAudioFormat method

The **GetDefaultExternalAudioFormat** method returns the default audio format for external audio recording.

Syntax

```
HRESULT     GetDefaultExternalAudioFormat (BMDScannerExternalAudioFormat
            *externalAudioFormat);
```

Parameters

Name	Direction	Description
`externalAudioFormat`	out	BMDScannerExternalAudioFormat can be 'EAAn' for external analog or 'EAAE' for AES.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.8 IScannerAudioSelect::SetSyncConfiguration method

The **SetSyncConfiguration** method sets the configuration of the Bi-phase/Timecode output.

Syntax

```
HRESULT     SetSyncConfiguration (BMDScannerSyncLTCConfig syncltc);
```

Parameters

Name	Direction	Description
syncltc	in	BMDScannerSyncLTCConfig to set.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.9 IScannerAudioSelect::GetSyncConfiguration method

The **GetSyncConfiguration** method returns the current configuration of the Bi-phase/Timecode output.

Syntax

```
HRESULT     GetSyncConfiguration (BMDScannerSyncLTCConfig *syncltc);
```

Parameters

Name	Direction	Description
syncltc	out	BMDScannerSyncLTCConfig specifying the current Bi-phase/Timecode configuration.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.14.10 IScannerAudioSelect::GetDefaultSyncConfiguration method

The **GetDefaultSyncConfiguration** method returns the default configuration of the Bi-phase/Timecode output.

Syntax

```
HRESULT    GetDefaultSyncConfiguration (BMDScannerSyncLTCConfig *syncltc);
```

Parameters

Name	Direction	Description
syncltc	out	BMDScannerSyncLTCConfig specifying the default Bi-phase/Timecode configuration.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15 IScannerReader Interface

The **IScannerReader** object interface represents the current **Cintel Audio and KeyKode Reader** configuration. Query it from the **IScanner** interface with **IID_IScannerReader** and then use the returned **IScannerReader** interface to access its methods described below.

Related Interfaces

Interface	Interface ID	Description
IScanner	IID_IScanner	Query the IID_IScannerReader interface from the IScanner interface. If a firmware update is required, this interface returns E_ACCESSDENIED when queried. Run the Cintel Scanner software utility to update the scanner firmware.

Public Member Functions	
Method	**Description**
GetReaderPresent	Return notification that the optional KeyKode Reader accesory is installed.
GetReaderMode	Return the keykode reader configuration for Cintel Audio and KeyKode Reader.
GetDefaultReaderMode	Return the default keykode reader configuration for Cintel Audio and KeyKode Reader.
SetReaderMode	Set the Cintel Audio and KeyKode Reader to record optical audio, optical audio narrow track, magnetic audio or KeyKode
GetMagneticAudioAllowed	Return whether magnetic audio is supported for the given film configuration.
SetAudioMode	Set the audio mode for Cintel Audio and KeyKode Reader to mono or stereo.
GetAudioMode	Return whether the Cintel Audio and KeyKode Reader to set to record mono or stereo.
GetDefaultAudioMode	Return whether the Cintel Audio and KeyKode Reader to set to record mono or stereo by default.
GetAudioPresent	Return if the audio is ready for capturing. This is determined by the capstan wheel movement and pre-roll delays required to be satisfied before the audio is frame accurate.

2.15.1 IScannerReader::GetReaderPresent method

The **GetReaderPresent** method confirms that the optional Cintel Audio and KeyKode Reader is installed and communicating properly and has the correct firmware loaded.

Syntax

```
HRESULT    GetReaderPresent (bool *present);
```

Parameters

Name	Direction	Description
present	out	Returns true if the Cintel Audio and KeyKode Reader is operational and ready for use, false otherwise. Observe notifiers for reasons of failure.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.2 IScannerReader::GetReaderMode method

The **GetReaderMode** method returns the configuration for Cintel Audio and KeyKode Reader if installed.

Syntax

```
HRESULT    GetReaderMode (BMDReaderMode *mode);
```

Parameters

Name	Direction	Description
mode	out	Returns the **BMDReaderMode** currently active.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.3 IScannerReader::GetDefaultReaderMode method

The **GetDefaultReaderMode** method returns the default keykode reader configuration for Cintel Audio and KeyKode Reader if installed.

Syntax

```
HRESULT     GetDefaultReaderMode (BMDReaderMode *mode);
```

Parameters

Name	Direction	Description
mode	out	**BMDReaderMode** specifying the default setting keykode reader configuration.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.4 IScannerReader::SetReaderMode method

The **SetReaderMode** method configures the Cintel Audio and KeyKode Reader if installed.

Syntax

```
HRESULT     SetReaderMode (BMDReaderMode mode);
```

Parameters

Name	Direction	Description
mode	in	Set the **BMDReaderMode** desired, such as audio or KeyKode.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.5 IScannerReader::GetMagneticAudioAllowed method

The **GetMagneticAudioAllowed** method returns whether magnetic stripe audio track is available for the current film configuration.

Syntax

```
HRESULT     GetMagneticAudioAllowed (bool *magneticAudioAllowed);
```

Parameters

Name	Direction	Description
`magneticAudioAllowed`	out	Returns true if the current film configuration supports magnetic audio, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.6 IScannerReader::SetAudioMode method

The **SetAudioMode** method configures the Cintel Audio and KeyKode Reader to mono or stereo.

Syntax

```
HRESULT     SetAudioMode (BMDReaderAudioMode audioMode);
```

Parameters

Name	Direction	Description
`audioMode`	in	Set the **BMDReaderAudioMode**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.7 IScannerReader::GetAudioMode method

The **GetAudioMode** method returns whether the Cintel Audio and KeyKode Reader to set to record mono or stereo.

Syntax

```
HRESULT     GetAudioMode (BMDReaderAudioMode *audioMode);
```

Parameters

Name	Direction	Description
`audioMode`	out	Returns the current **BMDReaderAudioMode**.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.8 IScannerReader::GetDefaultAudioMode method

The **GetDefaultAudioMode** method returns whether the Cintel Audio and KeyKode Reader to set to record mono or stereo by default.

Syntax

```
HRESULT    GetDefaultAudioMode (BMDReaderAudioMode *audioMode);
```

Parameters

Name	Direction	Description
audioMode	out	**BMDReaderAudioMode** specifying the default audio mode for recording.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

2.15.9 IScannerReader::GetAudioPresent method

The **GetAudioPresent** method returns whether the Cintel Audio and KeyKode Reader has audio valid and ready for capture. This depends on whether enough pre-roll has been applied, that the scanner is at the correct speed, and that the correct capture mode is entered such that the audio is stable and ready for capture.

Syntax

```
HRESULT    GetAudioPresent (bool *present);
```

Parameters

Name	Direction	Description
present	out	Returns true if the audio is able to be captured, false otherwise.

Return Values

Value	Description
E_FAIL	Failure
S_OK	Success

Section 3 — Common Data Types

3.1 Basic Types

`boolean`

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

Windows	`BOOL`
Mac	`bool`
Linux	`bool`

`Strings`

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

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

`int64_t`

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

Windows	`LONGLONG`
Mac	`int64_t`
Linux	`int64_t`

`uint64_t`

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

Windows	`ULONGLONG`
Mac	`uint64_t`
Linux	`uint64_t`

`uint32_t`

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

Windows	`unsigned int`
Mac	`uint32_t`
Linux	`uint32_t`

`int32_t`

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

Windows	`int`
Mac	`int32_t`
Linux	`int32_t`

uint16_t

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

Windows	`unsigned short`
Mac	`uint16_t`
Linux	`uint16_t`

uint8_t

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

Windows	`unsigned char`
Mac	`uint8_t`
Linux	`uint8_t`

3.2 Cintel Scanner API Information ID

BMDScannerAPIInformationID enumerates Cintel Scanner API identification information.

BMDScannerAPIVersion is a 32-bit unsigned integer that represents the Cintel Scanner API version number.

3.3 Display Modes

BMDScannerDisplayMode enumerates the video modes supported for output and input.

Mode	Width	Height	Frames per Second	Fields per Frame	Suggested Time Scale	Display Duration
bmdScannerModeCintelRAW	4096	3072	24	1	24000	1000
bmdScannerModeCintelCompressedRAW	4096	3072	24	1	24000	1000

NOTE The width and height for **bmdScannerModeCintelRAW** and **bmdScannerModeCintelCompressedRAW** display modes refer to the maximum uncropped frame resolution. Refer to the **bmdScannerFrameMetadataCintelImageWidth** and **bmdScannerFrameMetadataCintelImageHeight** Metadata IDs for the actual frame resolutions.

3.4 Pixel Formats

BMDScannerPixelFormat enumerates the pixel formats supported for output and input.

bmdScannerFormatUnspecified

Null indicates that the pixel format is not specified.

bmdScannerFormat12BitRAWGRBG

'r12p' This pixel format represents 12-bit RAW data for bayer pattern GRBG.

bmdScannerFormat12BitRAWJPEG

'r16p' This pixel format represents 12-bit RAW data arranged in tiles and JPEG compressed.

bmdScannerFormat12BitRAWDeprecated

'v210' This pixel format represents uncompressed data. It is important to note that this pixel format is deprecated as it is for DV9 files only.

3.5 Frame Flags

BMDScannerFrameFlags enumerates a set of flags applicable to a video frame.

bmdScannerFrameFlagDefault
Frame has default properties.

bmdScannerFrameFlagFlipVertical
Frame is inverted.

bmdScannerFrameContainsHDRMetadata
Frame contains metadata for HDR.

bmdScannerFrameContainsCintelMetadata
Frame contains Cintel metadata (see **BMDScannerFrameMetadataID**).

bmdScannerFrameHasNoInputSource
Frame contains no input source.

3.6 Audio Sample Rate

BMDScannerAudioSampleRate enumerates the possible audio sample rates.

bmdScannerAudioSampleRate48kHz
48 kHz sample rate

3.7 Audio Sample Types

BMDScannerAudioSampleType enumerates the possible audio sample types.

bmdScannerAudioSampleType16bitInteger
16 bit audio sample

bmdScannerAudioSampleType32bitInteger
32 bit audio sample

3.8 BMDScannerAudioSelectSource

BMDScannerAudioSelectSource is a 32-bit unsigned integer that represents which audio source is selected for capture of film audio.

bmdScannerAudioSourceNone
'ASNo' No audio source.

bmdScannerAudioSourceReader
'ASRe' Audio source is the Audio and KeyKode Reader.

bmdScannerAudioSourceExternal
'ASEx' Audio source is external, attached to an XLR audio input. To specify what sort of external audio input is used, use BMDScannerExternalAudioFormat.

3.9 BMDScannerExternalAudioFormat

BMDScannerExternalAudioFormat is a 32-bit unsigned integer that represents the type of external audio input.

bmdScannerExternalAudioAnalog

'EAAn' External audio is in analog format.

bmdScannerExternalAudioAES

'EAAE' External audio is in AES format.

3.10 BMDScannerTimecodeBCD

BMDScannerTimecodeBCD is a 32-bit unsigned integer that represents timecode in binary coded decimal format. 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

3.11 BMDScannerTimeValue

BMDScannerTimeValue is a 64-bit signed integer that represents hardware timing in units of **BMDScannerTimeScale**.

3.12 BMDScannerTimeScale

BMDScannerTimeScale is a 64-bit signed integer that represents the time scale for hardware timing in ticks per second.

3.13 Frame Metadata ID

BMDScannerFrameMetadataID enumerates the set of video frame metadata which may be queried from the **IScannerInputFrame** interface.

Cintel Metadata ID

Name	Type	Description
bmdScannerFrameMetadataCintelFlipRequired	bool	'cflr' The image should be flipped horizontally
bmdScannerFrameMetadataCintelFocusAssistEnabled	bool	'cfae' Focus Assist is currently enabled
bmdScannerFrameMetadataCintelInversionRequired	bool	'cinv' The image should be color inverted
bmdScannerFrameMetadataCintelFilmType	Int	Current film type
bmdScannerFrameMetadataCintelFilmGauge	Int	Current film gauge
bmdScannerFrameMetadataCintelKeykodeLow	Int	Raw keykode value - low 64 bits
bmdScannerFrameMetadataCintelKeykodeHigh	Int	Raw keykode value - high 64 bits
bmdScannerFrameMetadataCintelTile1Size	Int	Size in bytes of compressed raw tile 1
bmdScannerFrameMetadataCintelTile2Size	Int	Size in bytes of compressed raw tile 2
bmdScannerFrameMetadataCintelTile3Size	Int	Size in bytes of compressed raw tile 3
bmdScannerFrameMetadataCintelTile4Size	Int	Size in bytes of compressed raw tile 4
bmdScannerFrameMetadataCintelImageWidth	Int	Width in pixels of image
bmdScannerFrameMetadataCintelImageHeight	Int	Height in pixels of image
bmdScannerFrameMetadataCintelFilmFrameRate	Int	Film frame rate
bmdScannerFrameMetadataCintelLinearMaskingRedInRed	Float	Red in red linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingGreenInRed	Float	Green in red linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingBlueInRed	Float	Blue in red linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingRedInGreen	Float	Red in green linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingGreenInGreen	Float	Green in green linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingBlueInGreen	Float	Blue in green linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingRedInBlue	Float	Red in blue linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingGreenInBlue	Float	Green in blue linear masking parameter
bmdScannerFrameMetadataCintelLinearMaskingBlueInBlue	Float	Blue in blue linear masking parameter
bmdScannerFrameMetadataCintelLogMaskingRedInRed	Float	Red in red log masking parameter
bmdScannerFrameMetadataCintelLogMaskingGreenInRed	Float	Green in red log masking parameter
bmdScannerFrameMetadataCintelLogMaskingBlueInRed	Float	Blue in red log masking parameter
bmdScannerFrameMetadataCintelLogMaskingRedInGreen	Float	Red in green log masking parameter
bmdScannerFrameMetadataCintelLogMaskingGreenInGreen	Float	Green in green log masking parameter
bmdScannerFrameMetadataCintelLogMaskingBlueInGreen	Float	Blue in green log masking parameter

Name	Type	Description
bmdScannerFrameMetadataCintelLogMaskingRedInBlue	Float	Red in blue log masking parameter
bmdScannerFrameMetadataCintelLogMaskingGreenInBlue	Float	Green in blue log masking parameter
bmdScannerFrameMetadataCintelLogMaskingBlueInBlue	Float	Blue in blue log masking parameter
bmdScannerFrameMetadataCintelOffsetToApplyHorizontal	Float	Horizontal offset (pixels) to be applied to image
bmdScannerFrameMetadataCintelOffsetToApplyVertical	Float	Vertical offset (pixels) to be applied to image
bmdScannerFrameMetadataCintelSkewToApply	Int	Skew (pixels) to be applied to image
bmdScannerFrameMetadataCintelGainRed	Float	Red gain parameter to apply after log
bmdScannerFrameMetadataCintelGainGreen	Float	Green gain parameter to apply after log
bmdScannerFrameMetadataCintelGainBlue	Float	Blue gain parameter to apply after log
bmdScannerFrameMetadataCintelLiftRed	Float	Red lift parameter to apply after log and gain
bmdScannerFrameMetadataCintelLiftGreen	Float	Green lift parameter to apply after log and gain
bmdScannerFrameMetadataCintelLiftBlue	Float	Blue lift parameter to apply after log and gain
bmdScannerFrameMetadataCintelHDRGainRed	Float	Red gain parameter to apply to linear data for HDR Combine
bmdScannerFrameMetadataCintelHDRGainGreen	Float	Green gain parameter to apply to linear data for HDR Combine
bmdScannerFrameMetadataCintelHDRGainBlue	Float	Blue gain parameter to apply to linear data for HDR Combine

3.14 BMDScannerTransportCommand

BMDScannerTransportCommand enumerates transport commands for a scanner device.

Name	Type	Description
bmdScannerCommandStop	uint32_t	‘Stop’ command
bmdScannerCommandRunForward	uint32_t	‘RFwd’ command
bmdScannerCommandRunReverse	uint32_t	‘RRev’ command
bmdScannerCommandFastForward	uint32_t	‘FFwd’ command
bmdScannerCommandFastReverse	uint32_t	‘FRev’ command
bmdScannerCommandInchForward	uint32_t	‘IFwd’ command
bmdScannerCommandInchReverse	uint32_t	‘IRev’ command
bmdScannerCommandPerfForward	uint32_t	‘PFwd’ command
bmdScannerCommandPerfReverse	uint32_t	‘PRev’ command
bmdScannerCommandFramingUp	uint32_t	‘FrUp’ command
bmdScannerCommandFramingDown	uint32_t	‘FrDn’ command
bmdScannerCommandSlowForward	uint32_t	‘SlwF’ command
bmdScannerCommandSlowReverse	uint32_t	‘SlwR’ command

3.14.1 BMDScannerFilmType

BMDScannerFilmType enumerates film types for a scanner device.

Name	Type	Description
bmdScannerFilmTypeNegative	uint32_t	'FNEG' Negative film
bmdScannerFilmTypePositive	uint32_t	'FPOS' Positive fim
bmdScannerFilmTypeFloaterNegative	uint32_t	'FINE' Floaternegative film
bmdScannerFilmTypeFloaterPositive	uint32_t	'FIPO' Floaterpositive film
bmdScannerFilmTypeBWNegative	uint32_t	'FBNE' B/W Negative Film
bmdScannerFilmTypeBWPositive	uint32_t	'FBPO' B/W Positive Film
bmdScannerFilmTypeBWInterNegative	uint32_t	'FBIN' B/W InterNegative Film
bmdScannerFilmTypeBWInterPositive	uint32_t	'FBIP' B/W Interpsoitive Film

3.14.2 BMDScannerFilmGauge

BMDScannerFilmGauge enumerates film gauges for a scanner device.

Name	Type	Description
BMDScannerFilmGauge8mm	uint32_t	'8mmm' 8mm film
BMDScannerFilmGaugeSuper8mm	uint32_t	'S8mm' S8mm film
bmdScannerFilmGauge16mm	uint32_t	'16MM' 16mm film
bmdScannerFilmGauge35mm2Perf	uint32_t	'35M2' 2 perf 35mm film
bmdScannerFilmGauge35mm3Perf	uint32_t	'35M3' 3 perf 35mm film
bmdScannerFilmGauge35mm4Perf	uint32_t	'35M4' 4 perf 35mm film

3.14.3 BMDScannerWindType

BMDScannerWindType enumerates wind types for a scanner device.

Name	Type	Description
bmdScannerWindTypeA	uint32_t	'AWnd' A Wind
bmdScannerWindTypeB	uint32_t	'BWnd' B Wind

3.14.4 BMDScannerCompressionType

BMDScannerCompressionType enumerates the compression applied to a scanned clip.

Name	Type	Description
bmdScannerCompressionLossless	uint32_t	'CmLl' LossLess JPEG compression
bmdScannerCompressionLossy3To1	uint32_t	'C321' Lossy JPEG compression
bmdScannerCompressionNone	uint32_t	'CmNn' No compression

3.14.5 BMDScannerRollType

BMDScannerRollType enumerates the type of reel.

Name	Type	Description
bmdScannerRollTypeCore	uint32_t	'Core'
bmdScannerRollTypeReel	uint32_t	'Reel'

3.14.6 BMDScannerState

BMDScannerState enumerates scanner states.

Name	Type	Description
bmdScannerStateError	uint32_t	'SErr' Error
bmdScannerStateUnloaded	uint32_t	'SUld' Unloaded
bmdScannerStateLoading	uint32_t	'SLdg' Loading
bmdScannerStateFramingUp	uint32_t	'SFrU' Framing Up
bmdScannerStateFramingDown	uint32_t	'SFrD' Framing Down
bmdScannerStatePerfNudgingUp	uint32_t	'SPNU' Nudging Up
bmdScannerStatePerfNudgingDown	uint32_t	'SPND' Nudging Down
bmdScannerStateInchingForward	uint32_t	'SIFw' Inching Forward
bmdScannerStateInchingReverse	uint32_t	'SIRv' Inching Reverse
bmdScannerStateStopped	uint32_t	'SLdd' Stopped
bmdScannerStateRunForwards	uint32_t	'SFwd' Run Forward
bmdScannerStateRunReverse	uint32_t	'SRev' Run Reverse
bmdScannerStateShuttleForwards	uint32_t	'SSFw' Shuttle Forward
bmdScannerStateShuttleReverse	uint32_t	'SSRv' Shuttle Reverse
bmdScannerStateCueingForward	uint32_t	'SCFw' Cueing Forward
bmdScannerStateCueingReverse	uint32_t	'SCRv' Cueing Reverse
bmdScannerStateStopping	uint32_t	'SStg' Stopping
bmdScannerStateUnloading	uint32_t	'SUnl' Unloading
bmdScannerStateLEDCalibrating	uint32_t	'SCLD' Calibrating LEDs
bmdScannerStateCalibratingFixedPatternGain	uint32_t	'SCFG' Calibrating FPG
bmdScannerStateSleeping	uint32_t	'SSlp' Sleeping
bmdScannerStateCapturing	uint32_t	'SCap' Capturing
bmdScannerStateCapturingHDRForward	uint32_t	'SCHF' HDR Capturing Forward

3.14.7 BMDScannerMessage

BMDScannerMessage enumerates scanner errors, warnings, and general information messages.

Name	Type	Description
bmdScannerUnused	uint32_t	'ZZZZ'
bmdScannerMessageAmplifierOverTemperature	uint32_t	'HOvT' Amplifier Over Temperature
bmdScannerMessageAmplifierError	uint32_t	'HAmE' Amplifier Error
bmdScannerMessageAmplifierOverCurrent	uint32_t	'AmpC' Amplifier Over Current
bmdScannerMessagePulserError	uint32_t	'PlsE' Pulser Error
bmdScannerMessagePulserOverCurrent	uint32_t	'PlsC' Pulser Over Current
bmdScannerMessageFPGAOverTemperature	uint32_t	'FPGT' FPGA Over Temperature
bmdScannerMessageFPGAFanFail	uint32_t	'FPFF' FPGA Fan Fail
bmdScannerMessageSensorTemperatureOutOfRange	uint32_t	'SnsT' Sensor Over Temperature
bmdScannerMessageSensorFanFail	uint32_t	'SnfF' Sensor Fan Fail
bmdScannerMessageFeedSlipping	uint32_t	'SFVe' Feed Slipping
bmdScannerMessageTakeupSlipping	uint32_t	'STVe' Takeup Slipping
bmdScannerMessageTransportError	uint32_t	'TnsE' Transport Error
bmdScannerMessageStepperSlipping	uint32_t	'StSl' Stepper Slipping
bmdScannerMessageMagHeadEngaged	uint32_t	'MgDs' Mag Head Engaged Error
bmdScannerMessageMagHeadDisengaged	uint32_t	'MgEn' Mag Head Disengage Error
bmdScannerMessageProgrammingStarted	uint32_t	'Prgr' Programming In Progress
bmdScannerMessageReaderIncompatible	uint32_t	'RdIn' Reader Incompatible
bmdScannerMessageCodecError	uint32_t	'CODE' Codec Error
bmdScannerMessageHDRUnavailable	uint32_t	'NoHD' HDR Unavailable
bmdScannerMessageImageClipped	uint32_t	'ImCl' Image In Clip
bmdScannerMessageInfillClipped	uint32_t	'InCl' Perf Infill In Clip
bmdScannerMessageFilmConfigurationChanged	uint32_t	'FICf' Film Configuration Changed
bmdScannerMessageTransportParametersChanged	uint32_t	'TpPa' Transport Parameter Changed
bmdScannerMessageColorPropertiesChanged	uint32_t	'ClPp' Color Properties Changed
bmdScannerMessageImageStabilizerChanged	uint32_t	'ImSt' Image Stabilizer Changed
bmdScannerMessageReaderSettingsChanged	uint32_t	'RdSt' Reader Settings Changed
bmdScannerMessageCaptureSettingsChanged	uint32_t	'CpSt' Capture Settings Changed
bmdScannerMessageAudioSelectSettingsChanged	uint32_t	'ASSt' Audio Select Settings Changed

3.14.8 BMDScannerMessageSeverity

BMDScannerMessageSeverity enumerates severity levels for scanner messages.

Name	Type	Description
bmdScannerMessageSeverityError	uint32_t	'Eror' Error
bmdScannerMessageSeverityWarning	uint32_t	'Warn' Warning
bmdScannerMessageSeverityInformation	uint32_t	'Info' Information only

3.14.9 BMDScannerCalibrationStatus

BMDScannerCalibrationStatus enumerates the calibration status.

Name	Type	Description
bmdScannerCalibrationNotRun	uint32_t	'NtRn' Not Yet Run
bmdScannerCalibrationRunning	uint32_t	'Rnng' Running
bmdScannerCalibrationComplete	uint32_t	'Cmpl' Complete
bmdScannerCalibrationCompromised	uint32_t	'Cmpr' Complete but compromised. This can happen with very darkly toned film.
bmdScannerCalibrationFailed	uint32_t	'Flld' Failed. Calibration is unable to achieve an acceptable light balance.

3.15 BMDReaderMode

BMDReaderMode enumerates the KeyKode audio reader mode.

Name	Type	Description
bmdReaderModeKeyKodePerfs	uint32_t	Counts the number of perforations between KeyKode counts.
bmdReaderModeKeyKodeFrames	uint32_t	Counts from 0 to 15 between updates for 35mm 4 perf, 0 to 11 for 35mm 3 perf, 0 to 7 for 35mm 2 perf, and 0 to 3 for 16mm film. This is because KeyKodes appear every 4 frames multiplied by the number of perforations of the given film gauge.
bmdReaderModeAudio	uint32_t	'MdAu' Audio Wide Slit mode
bmdReaderModeAudioNarrowSlit	uint32_t	'MdAn' Audio Narrow Slit mode, for shrunken material
bmdReaderModeAudioMagnetic	uint32_t	'MdAM' Audio Magnetic mode

3.16 BMDReaderAudioMode

BMDReaderAudioMode enumerates the reader audio mode.

Name	Type	Description
bmdReaderAudioModeMono	uint32_t	'AMMo' Mono. Stereo channels combined and output on both channels.
bmdReaderAudioModeStereo	uint32_t	'AMSt' Stereo. Both audio channels are preserved separately.

3.17 BMDScannerCaptureMode

BMDScannerCaptureMode enumerates the capture mode.

Name	Type	Description
bmdScannerCaptureModePreview	uint32_t	'CpNn' Preview
bmdScannerCaptureModeNormal	uint32_t	'CpNl' Normal Exposure
bmdScannerCaptureModeHigh	uint32_t	'CpHi' High Exposure

3.18 BMDScannerAdvancedFlag

BMDScannerAdvancedFlag enumerates the Advanced flag.

Name	Type	Description
bmdScannerFlagsFullStepper	uint32_t	= 1 << 0 Puts the driver motor at full power all the time. This can sound worse but it may be required for some film types such as Teflon.
bmdScannerFlagsRunRewind	uint32_t	= 1 << 2 Rather than stop the film at the end of the reel, it will automatically rewind.

3.19 BMDScannerSyncLTCConfig

BMDScannerSyncLTCConfig enumerates Bi-Phase Sync or LTC output select and configuration.

Name	Type	Description
bmdScannerSync1ppf	uint32_t	'S1pp' Bi-phase Sync at 1 pulse per frame
bmdScannerSync2ppf	uint32_t	'S2pp' Bi-phase Sync at 2 pulses per frame
bmdScannerSync5ppf	uint32_t	'S5pp' Bi-phase Sync at 5 pulses per frame
bmdScannerSync10ppf	uint32_t	'SApp' Bi-phase Sync at 10 pulses per frame
bmdScannerLTCFrameSync	uint32_t	'LTFS' Linear TimeCode is output once per frame
bmdScannerLTCFreeRun	uint32_t	'LTFR' Linear TimeCode is output at the film frame rate, regardless of transport speed
bmdScannerSyncLTCUnsupported	uint32_t	'UnSp' Unsupported feature