IsoSDK dotNet Documentation

Version 10.3.0

Table of Contents

Index

1 Symbol Reference

1.1 IsoSDK Namespace

This is namespace IsoSDK.

Classes

	Name	Description
	AddFileEventArgs (see page 5)	Provides data and arguments for the AddFile event.
	AudioDecodeDoneEventArgs (see page 7)	Provides data and arguments for the AudioDecodeDone event.
	AudioDecoderEventArgs (see page 8)	Provides data and arguments for the AudioDecoder event.
	BootOptions (see page 10)	This class will used to manage BootOptions information. A instance of this class will created on call of the Property Burner::BootOptions.
	BurnDoneEventArgs (see page 12)	Provides data and arguments for the BurnDone event.
	Burner (see page 13)	This is class IsoSDK::Burner.
	BurnFileEventArgs (see page 87)	Provides data and arguments for the BurnFile event.
	CDText (see page 87)	This class will used to manage CD-Text information. The FoxBurner SDK can reaad CD-Text of the disk and a give track index. A instance of this class will created on call of the method ReadCDText.
	CompareFilesForArrangementEventArgs (see page 89)	Provides data and arguments for the CompareFilesForArrangement event.
	CompressEncryptOptions (see page 90)	This class will used to manage the CompressEncryptOptions information. A instance of this class will created on call of the Property CompressEncryptOptions.
	CreateDirEventArgs (see page 92)	Provides data and arguments for the CreateDir event.
	DiskDirectory (see page 93)	This class will used to manage a directroy of a DiskSession (see page 94) object. A instance of this class will created on call of the method OpenDirectory.
	DiskSession (see page 94)	This class will used to manage a session of a Burner (see page 13) object. A instance of this class will created on call of the method OpenDiskSession.
	DVDVideoOptions (see page 98)	This class will used to manage the DVDVideoOptions information. A instance of this class will created on call of the Property DVDVideoOptions.
	EraseDoneEventArgs (see page 99)	Provides data and arguments for the EraseDone event.
	ExtendedDeviceCapabilities (see page 100)	This class will used to manage the extended device capabilities information. A instance of this class will created on call of the method GetDeviceInformationEx.
	FileDateTime (see page 101)	A structure that contains the information about the time & date of the specific file.

new	FileDateTimeEx (see page 104)	This class is used to manage the custom date / time values for files, directory and file system entries. With this values you can overwrite the common date / time values of the files. You can use the system time or own time values. This date / time value is set global but you can overwrite each file value with SetFileTimes method. This settings will not changed old added files / directories only those that get added after calling this. Will get reset with define a new project.
	GeneralOptions (see page 107)	This class will used to manage the GeneralOptions information. A instance of this class will created on call of the Property Options.
	InfoTextEventArgs (see page 113)	Provides data and arguments for the InfoText event.
	ISOExOptions (see page 114)	This class will used to manage ISOExOptions information. A instance of this class will created on call of the Property ISOExOptions.
	NetworkTags (see page 122)	This class will used to manage NetworkTags information. The NetWorkTags are the information of the Disk/Track received from a CDDB/FreeDB database.
	ProcessEventArgs (see page 124)	Provides data and arguments for the Process event.
	RemoveFileEventArgs (see page 126)	Provides data and arguments for the RemoveFile event.
	TextEventArgs (see page 127)	Provides data and arguments for the Text event.
	UDFOptions (see page 128)	This class will used to manage UDFOptions information. A instance of this class will created on call of the Property UDFOptions.
	VerifyDoneEventArgs (see page 130)	Provides data and arguments for the VerifyDone event.
	VerifyErrorEventArgs (see page 131)	Provides data and arguments for the VerifyError event.
	VerifyFileEventArgs (see page 132)	Provides data and arguments for the VerifyFile event.
	VerifySectorEventArgs (see page 133)	Provides data and arguments for the VerifySector event.
	VideoScanDoneEventArgs (see page 134)	Provides data and arguments for the VideoScanDone event.
	VideoScannerEventArgs (see page 137)	Provides data and arguments for the VideoScanner event.

Structs, Records, Enums

	Name	Description
	AspectRatio (see page 140)	This enumeration defines the possible Aspect Ratios of a MPEG Movie the IsoSDK supports.
	ASPIInterface (see page 141)	This enumeration defines the ASPI interfaces the IsoSDK supports.
	AudioFileProperty (see page 141)	This structure contains information about CD-Text.
	AudioFormat (see page 142)	This enumeration defines the audio formats the IsoSDK supports for encoding.
	AudioGrabbingParams (see page 143)	A structure that contains information for Audiograbbing
	BitrateType (see page 144)	This enumeration defines the bitrate types the IsoSDK supports.
	BootVolumeInfo (see page 144)	A structure that contains the information about the boot volume information.
	BurnIsoOptions (see page 145)	A structure that contains the information that are needed to burn an ISO file.
	Capabilities (see page 146)	This enumeration defines the capabilities of the device. Please click on the feature you want to get information for.
	CDTextContentItem (see page 151)	This enumeration defines the types of CD-Text information the IsoSDK supports.
	CreateImageParams (see page 152)	A structure that contains information for the ImageCreate operation.
	DeviceIndex (see page 153)	This enumeration defines the possible selection for devices.

	DeviceInformation (see page 153)	A structure that contains device information of a device from the internal device list.
new	DiscSignature (see page 154)	This enumeration defines the compression and/or encryption state of the disc or image.
	DiskCopyOptions (see page 154)	A structure that contains information for the disk copy operation.
	ErrorCode (see page 155)	This enumeration defines the error codes the IsoSDK can throw out.
	ExtendedDeviceInformation (see page 181)	A structure that contains the extended information for the device.
	ExtendedMediumType (see page 182)	This enumeration defines the extended type of the current medium.
	Extent (see page 183)	A structure that contains the information about the Extent information of a file in allocation table.
	FileAllocationTable (see page 183)	A structure that contains the information about the file allocation table.
	FileAttributes (see page 184)	This enumeration defines the file attribute of a file to add.
	FileEntry (see page 185)	A structure that contains the information for the file to be used.
	FileSystems (see page 186)	This enumeration defines the file system of a medium the IsoSDK supports.
	ImageFormat (see page 186)	This enumeration defines the image formats the IsoSDK supports for writing.
	ImageTask (see page 187)	This enumeration defines the possible actions for image creation task while disk copy.
	InfoLevel (see page 187)	This enumeration defines the level of informations that the IsoSDK supports.
	ISOLevel (see page 188)	This enumeration defines the possible extended ISO9660 derivate the IsoSDK supports.
	ISOVolumeInfo (see page 189)	A structure that contains the information about the ISO information.
	MediumInfo (see page 190)	A structure that contains the information about the medium.
	MediumStatus (see page 191)	This enumeration defines the status of the current medium.
	MediumType (see page 192)	This enumeration defines the type of the current medium.
	NetworkTagsContentItem (see page 194)	This enumeration defines the CDDB fields that are supported by the IsoSDK.
	ProjectType (see page 195)	This enumeration defines the project types the IsoSDK supports.
	RawDataType (see page 196)	This enumeration defines the data type for a RAW Project type.
	RawTrack (see page 196)	This structure contains the information of a RAW image.
	RawTrackFormat (see page 197)	This enumeration defines the track format type for a RAW project type.
	ReadErrorCorrectionParams (see page 198)	A structure that contains the information of the ReadErrorCorrection for CopyDisk and CreateImage.
	ReadMode (see page 198)	This enumeration defines the read mode the IsoSDK supports.
	SavePathOption (see page 199)	This enumeration defines if and how the super ordinate directory is added as a folder to the project.
	SaveTrackFileFormat (see page 200)	This enumeration defines the possible formats the IsoSDK supports to save tracks.
	SessionInfo (see page 200)	A structure that contains the information about the selected session.
	SessionStatus (see page 201)	This enumeration defines the status of the last session.
	Speed (see page 201)	This structure contains information about the possible burning speeds.
	TagChoiceType (see page 202)	This enumeration defines the type of tags the IsoSDK will try to receive.
	TrackFormat (see page 202)	This enumeration defines the track type of a medium.
	TrackInfo (see page 203)	A structure that contains the information about the selected track.
	UDFPartitionType (see page 204)	This enumeration defines the UDF partition the IsoSDK supports.
	UDFVersion (see page 205)	This enumeration defines the UDF version the IsoSDK supports.
	UDFVolumeInfo (see page 206)	A structure that contains the information about the UDF information.
	WriteMethod (see page 207)	This enumeration defines the write methods the IsoSDK supports.

1.1.1 Classes

The following table lists classes in this documentation.

Classes

	Name	Description
	AddFileEventArgs (see page 5)	Provides data and arguments for the AddFile event.
	AudioDecodeDoneEventArgs (see page 7)	Provides data and arguments for the AudioDecodeDone event.
	AudioDecoderEventArgs (see page 8)	Provides data and arguments for the AudioDecoder event.
	BootOptions (see page 10)	This class will used to manage BootOptions information. A instance of this class will created on call of the Property Burner::BootOptions.
	BurnDoneEventArgs (see page 12)	Provides data and arguments for the BurnDone event.
	Burner (see page 13)	This is class IsoSDK::Burner.
	BurnFileEventArgs (see page 87)	Provides data and arguments for the BurnFile event.
	CDText (see page 87)	This class will used to manage CD-Text information. The FoxBurner SDK can reaad CD-Text of the disk and a give track index. A instance of this class will created on call of the method ReadCDText.
	CompareFilesForArrangementEventArgs (see page 89)	Provides data and arguments for the CompareFilesForArrangement event.
	CompressEncryptOptions (see page 90)	This class will used to manage the CompressEncryptOptions information. A instance of this class will created on call of the Property CompressEncryptOptions.
	CreateDirEventArgs (see page 92)	Provides data and arguments for the CreateDir event.
	DiskDirectory (see page 93)	This class will used to manage a directroy of a DiskSession (see page 94) object. A instance of this class will created on call of the method OpenDirectory.
	DiskSession (see page 94)	This class will used to manage a session of a Burner (see page 13) object. A instance of this class will created on call of the method OpenDiskSession.
	DVDVideoOptions (see page 98)	This class will used to manage the DVDVideoOptions information. A instance of this class will created on call of the Property DVDVideoOptions.
	EraseDoneEventArgs (see page 99)	Provides data and arguments for the EraseDone event.
	ExtendedDeviceCapabilities (see page 100)	This class will used to manage the extended device capabilities information. A instance of this class will created on call of the method GetDeviceInformationEx.
new	FileDateTimeEx (see page 104)	This class is used to manage the custom date / time values for files, directory and file system entries. With this values you can overwrite the common date / time values of the files. You can use the system time or own time values. This date / time value is set global but you can overwrite each file value with SetFileTimes method. This settings will not changed old added files / directories only those that get added after calling this. Will get reset with define a new project.
	GeneralOptions (see page 107)	This class will used to manage the GeneralOptions information. A instance of this class will created on call of the Property Options.
	InfoTextEventArgs (see page 113)	Provides data and arguments for the InfoText event.
	ISOExOptions (see page 114)	This class will used to manage ISOExOptions information. A instance of this class will created on call of the Property ISOExOptions.

	NetworkTags (see page 122)	This class will used to manage NetworkTags information. The NetWorkTags are the information of the Disk/Track received from a CDDB/FreeDB database.
	ProcessEventArgs (see page 124)	Provides data and arguments for the Process event.
	RemoveFileEventArgs (see page 126)	Provides data and arguments for the RemoveFile event.
	TextEventArgs (see page 127)	Provides data and arguments for the Text event.
	UDFOptions (see page 128)	This class will used to manage UDFOptions information. A instance of this class will created on call of the Property UDFOptions.
	VerifyDoneEventArgs (see page 130)	Provides data and arguments for the VerifyDone event.
	VerifyErrorEventArgs (see page 131)	Provides data and arguments for the VerifyError event.
	VerifyFileEventArgs (see page 132)	Provides data and arguments for the VerifyFile event.
	VerifySectorEventArgs (see page 133)	Provides data and arguments for the VerifySector event.
	VideoScanDoneEventArgs (see page 134)	Provides data and arguments for the VideoScanDone event.
	VideoScannerEventArgs (see page 137)	Provides data and arguments for the VideoScanner event.

Structures

	Name	Description
	FileDateTime (see page 101)	A structure that contains the information about the time & date of the specific file.

1.1.1.1 AddFileEventArgs Class

Class Hierarchy

C++

```
ref class AddFileEventArgs : public EventArgs;
```

C#

```
public class AddFileEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the AddFile event.

1.1.1.1.1 AddFileEventArgs Members

The following tables list the members exposed by AddFileEventArgs.

Public Data Members

	Name	Description
	m_dFileDateTime (see page 6)	Timestamp of the last change.
	m_dFileSize (see page 6)	A double with the file size in bytes.
	m_strFullPath (see page 6)	A string with the source path of the added file (includes file name).
	m_strISOName (see page 6)	A string with the ISO9660 name of the added file.
	m_strJolietName (see page 7)	A string with the Joliet name of the added file.
	m_strUDFName (see page 7)	A string with the UDF name of the added file.

1.1.1.1.2 AddFileEventArgs Data Members

The data members of the AddFileEventArgs class are listed here.

Public Data Members

	Name	Description
	m_dFileDateTime (see page 6)	Timestamp of the last change.
	m_dFileSize (see page 6)	A double with the file size in bytes.
	m_strFullPath (see page 6)	A string with the source path of the added file (includes file name).
	m_strISOName (see page 6)	A string with the ISO9660 name of the added file.
	m_strJolietName (see page 7)	A string with the Joliet name of the added file.
	m_strUDFName (see page 7)	A string with the UDF name of the added file.

1.1.1.1.2.1 AddFileEventArgs::m_dFileDateTime Data Member

C++

```
double m_dFileDateTime;
```

C#

```
public double m_dFileDateTime;
```

Description

Timestamp of the last change.

1.1.1.1.2.2 AddFileEventArgs::m_dFileSize Data Member

C++

```
double m_dFileSize;
```

C#

```
public double m_dFileSize;
```

Description

A double with the file size in bytes.

1.1.1.1.2.3 AddFileEventArgs::m_strFullPath Data Member

C++

```
String ^ m_strFullPath;
```

C#

```
public String m_strFullPath;
```

Description

A string with the source path of the added file (includes file name).

1.1.1.1.2.4 AddFileEventArgs::m_strISOName Data Member

C++

```
String ^ m_strISOName;
```

C#

```
public String m_strISOName;
```

Description

A string with the ISO9660 name of the added file.

1.1.1.1.2.5 AddFileEventArgs::m_strJolietName Data Member

C++

```
String ^ m_strJolietName;
```

C#

```
public String m_strJolietName;
```

Description

A string with the Joliet name of the added file.

1.1.1.1.2.6 AddFileEventArgs::m_strUDFName Data Member

C++

```
String ^ m_strUDFName;
```

C#

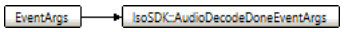
```
public String m_strUDFName;
```

Description

A string with the UDF name of the added file.

1.1.1.2 AudioDecodeDoneEventArgs Class

Class Hierarchy



C++

```
ref class AudioDecodeDoneEventArgs : public EventArgs;
```

C#

```
public class AudioDecodeDoneEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the AudioDecodeDone event.

1.1.1.2.1 AudioDecodeDoneEventArgs Members

The following tables list the members exposed by AudioDecodeDoneEventArgs.

Public Data Members

	Name	Description
	m_nErrorCode (see page 8)	Error ID when an error occurred. See Language file.
	m_strError (see page 8)	A string with the decoding error. If no error occurred, an empty string will be returned.
	m_strFileName (see page 8)	A string with the file name and path of the audio file that was decoded.

1.1.1.2.2 AudioDecodeDoneEventArgs Data Members

The data members of the AudioDecodeDoneEventArgs class are listed here.

Public Data Members

	Name	Description
	m_nErrorCode (see page 8)	Error ID when an error occurred. See Language file.
	m_strError (see page 8)	A string with the decoding error. If no error occurred, an empty string will be returned.
	m_strFileName (see page 8)	A string with the file name and path of the audio file that was decoded.

1.1.1.2.2.1 AudioDecodeDoneEventArgs::m_nErrorCode Data Member

C++

```
int m_nErrorCode;
```

C#

```
public int m_nErrorCode;
```

Description

Error ID when an error occurred. See Language file.

1.1.1.2.2.2 AudioDecodeDoneEventArgs::m_strError Data Member

C++

```
String ^ m_strError;
```

C#

```
public String m_strError;
```

Description

A string with the decoding error. If no error occurred, an empty string will be returned.

1.1.1.2.2.3 AudioDecodeDoneEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

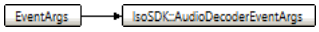
```
public String m_strFileName;
```

Description

A string with the file name and path of the audio file that was decoded.

1.1.1.3 AudioDecoderEventArgs Class

Class Hierarchy



C++

```
ref class AudioDecoderEventArgs : public EventArgs;
```

C#

```
public class AudioDecoderEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the AudioDecoder event.

1.1.1.3.1 AudioDecoderEventArgs Members

The following tables list the members exposed by AudioDecoderEventArgs.

Public Data Members

	Name	Description
	m_fPercent (see page 9)	A float with the decoding progress in % (0 – 100)
	m_nAudioType (see page 9)	Deprecated
	m_strFileName (see page 9)	A string with the file name and path of the audio file that was decoded.

1.1.1.3.2 AudioDecoderEventArgs Data Members

The data members of the AudioDecoderEventArgs class are listed here.

Public Data Members

	Name	Description
	m_fPercent (see page 9)	A float with the decoding progress in % (0 – 100)
	m_nAudioType (see page 9)	Deprecated
	m_strFileName (see page 9)	A string with the file name and path of the audio file that was decoded.

1.1.1.3.2.1 AudioDecoderEventArgs::m_fPercent Data Member

C++

```
float m_fPercent;
```

C#

```
public float m_fPercent;
```

Description

A float with the decoding progress in % (0 – 100)

1.1.1.3.2.2 AudioDecoderEventArgs::m_nAudioType Data Member

C++

```
int m_nAudioType;
```

C#

```
public int m_nAudioType;
```

Description

Deprecated

1.1.1.3.2.3 AudioDecoderEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

```
public String m_strFileName;
```

Description

A string with the file name and path of the audio file that was decoded.

1.1.1.4 BootOptions Class

Class Hierarchy

IsoSDK::BootOptions

C++

```
ref class BootOptions;
```

C#

```
public class BootOptions;
```

File

Options.h

Description

This class will used to manage BootOptions information. A instance of this class will created on call of the Property Burner::BootOptions.

1.1.1.4.1 BootOptions Members

The following tables list the members exposed by BootOptions.

Public Properties

	Name	Description
	BootIndicator (see page 11)	Get/Set the Byte 0 of the boot catalog. true = Bootable (0x88) and false = Not Bootable (0x00).
	DeveloperID (see page 11)	Get/Set the Byte 4-1F pf the Header Entry. It contains the string to the developer. E. g. your company name.
	Emulation (see page 11)	Get/Set the Byte 1 from the boot catalog. It sets the boot media type. It specifies what media the boot image is intended to emulate. 0 = No Emulation 1 = 1.2 MB diskette 2 = 1.44 MB diskette 3 = 2.88 MB diskette 4 = Hard Disk
	LoadSegment (see page 12)	Get/Set the Byte 2-3 of the boot catalog. The default value is 1984 (7C0). It is the address of the load segment for the initial boot image.
	PlatformID (see page 12)	The platform ID, Byte 1 of the header entry: 0 = 80x86 1 = Power PC 2 = Mac
	SectorCount (see page 12)	Get/Set the Byte 6-7 of the boot catalog. This is the number of virtual/emulated sectors the system will store at the load segment.

1.1.1.4.2 BootOptions Properties

The properties of the BootOptions class are listed here.

Public Properties

	Name	Description
	BootIndicator (see page 11)	Get/Set the Byte 0 of the boot catalog. true = Bootable (0x88) and false = Not Bootable (0x00).

	DeveloperID (see page 11)	Get/Set the Byte 4-1F pf the Header Entry. It contains the string to the developer. E. g. your company name.
	Emulation (see page 11)	Get/Set the Byte 1 from the boot catalog. It sets the boot media type. It specifies what media the boot image is intended to emulate. 0 = No Emulation 1 = 1.2 MB diskette 2 = 1.44 MB diskette 3 = 2.88 MB diskette 4 = Hard Disk
	LoadSegment (see page 12)	Get/Set the Byte 2-3 of the boot catalog. The default value is 1984 (7C0). It is the address of the load segment for the initial boot image.
	PlatformID (see page 12)	The platform ID, Byte 1 of the header entry: 0 = 80x86 1 = Power PC 2 = Mac
	SectorCount (see page 12)	Get/Set the Byte 6-7 of the boot catalog. This is the number of virtual/emulated sectors the system will store at the load segment.

1.1.1.4.2.1 BootOptions::BootIndicator Property

C++

```
property bool BootIndicator;
```

C#

```
public bool BootIndicator;
```

Description

Get/Set the Byte 0 of the boot catalog. true = Bootable (0x88) and false = Not Bootable (0x00).

1.1.1.4.2.2 BootOptions::DeveloperID Property

C++

```
property String ^ DeveloperID;
```

C#

```
public String DeveloperID;
```

Description

Get/Set the Byte 4-1F pf the Header Entry. It contains the string to the developer. E. g. your company name.

1.1.1.4.2.3 BootOptions::Emulation Property

C++

```
property int Emulation;
```

C#

```
public int Emulation;
```

Description

Get/Set the Byte 1 from the boot catalog. It sets the boot media type. It specifies what media the boot image is intended to emulate.

0 = No Emulation

1 = 1.2 MB diskette

2 = 1.44 MB diskette

3 = 2.88 MB diskette

4 = Hard Disk

1.1.1.4.2.4 BootOptions::LoadSegment Property

C++

```
property int LoadSegment;
```

C#

```
public int LoadSegment;
```

Description

Get/Set the Byte 2-3 of the boot catalog. The default value is 1984 (7C0). It is the address of the load segment for the initial boot image.

1.1.1.4.2.5 BootOptions::PlatformID Property

C++

```
property int PlatformID;
```

C#

```
public int PlatformID;
```

Description

The platform ID, Byte 1 of the header entry:

0 = 80x86

1 = Power PC

2 = Mac

1.1.1.4.2.6 BootOptions::SectorCount Property

C++

```
property int SectorCount;
```

C#

```
public int SectorCount;
```

Description

Get/Set the Byte 6-7 of the boot catalog. This is the number of virtual/emulated sectors the system will store at the load segment.

1.1.1.5 BurnDoneEventArgs Class

Class Hierarchy

C++

```
ref class BurnDoneEventArgs : public EventArgs;
```

C#

```
public class BurnDoneEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the BurnDone event.

1.1.1.5.1 BurnDoneEventArgs Members

The following tables list the members exposed by BurnDoneEventArgs.

Public Data Members

	Name	Description
	m_strError (see page 13)	A string with the burning error. If no error occurred, an empty string or a null pointer will be returned.

1.1.1.5.2 BurnDoneEventArgs Data Members

The data members of the BurnDoneEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strError (see page 13)	A string with the burning error. If no error occurred, an empty string or a null pointer will be returned.

1.1.1.5.2.1 BurnDoneEventArgs::m_strError Data Member

C++

```
String ^ m_strError;
```

C#

```
public String m_strError;
```

Description

A string with the burning error. If no error occurred, an empty string or a null pointer will be returned.

1.1.1.6 Burner Class

Class Hierarchy

IsoSDK::Burner

C++

```
ref class Burner;
```

C#

```
public class Burner;
```

File

IsoSDKBurnerNet.h

Description

This is class IsoSDK::Burner.

1.1.1.6.1 Burner Members

The following tables list the members exposed by Burner.

Public Delegates

Name	Description
AddFileEventHandler (see page 77)	This is nested type IsoSDK::Burner::AddFileEventHandler.
AudioDecodeDoneEventHandler (see page 78)	This is nested type IsoSDK::Burner::AudioDecodeDoneEventHandler.
AudioDecoderEventHandler (see page 78)	This is nested type IsoSDK::Burner::AudioDecoderEventHandler.
BurnDoneEventHandler (see page 78)	This is nested type IsoSDK::Burner::BurnDoneEventHandler.
BurnFileEventHandler (see page 78)	This is nested type IsoSDK::Burner::BurnFileEventHandler.
CompareFilesForArrangementHandler (see page 78)	This is nested type IsoSDK::Burner::CompareFilesForArrangementHandler.
CreateDirEventHandler (see page 79)	This is nested type IsoSDK::Burner::CreateDirEventHandler.
EraseDoneEventHandler (see page 79)	This is nested type IsoSDK::Burner::EraseDoneEventHandler.
FinalizeEventHandler (see page 79)	This is nested type IsoSDK::Burner::FinalizeEventHandler.
InfoTextEventHandler (see page 79)	This is nested type IsoSDK::Burner::InfoTextEventHandler.
JobDoneEventHandler (see page 79)	This is nested type IsoSDK::Burner::JobDoneEventHandler.
ProcessEventHandler (see page 80)	This is nested type IsoSDK::Burner::ProcessEventHandler.
RemoveFileEventHandler (see page 80)	This is nested type IsoSDK::Burner::RemoveFileEventHandler.
StartVerifyEventHandler (see page 80)	This is nested type IsoSDK::Burner::StartVerifyEventHandler.
TextEventHandler (see page 80)	This is nested type IsoSDK::Burner::TextEventHandler.
VerifyDoneEventHandler (see page 80)	This is nested type IsoSDK::Burner::VerifyDoneEventHandler.
VerifyErrorEventHandler (see page 81)	This is nested type IsoSDK::Burner::VerifyErrorEventHandler.
VerifyFileEventHandler (see page 81)	This is nested type IsoSDK::Burner::VerifyFileEventHandler.
VerifySectorEventHandler (see page 81)	This is nested type IsoSDK::Burner::VerifySectorEventHandler.
VideoScanDoneEventHandler (see page 81)	This is nested type IsoSDK::Burner::VideoScanDoneEventHandler.
VideoScannerEventHandler (see page 81)	This is nested type IsoSDK::Burner::VideoScannerEventHandler.

Public Events

	Name	Description
	AddFileEvent (see page 82)	This event will be triggered when a file was successfully added to the project.
	AudioDecodeDoneEvent (see page 83)	This event occurs when the process of decoding was completed.
	AudioDecoderEvent (see page 83)	This event will be triggered while the process of decoding.
	BurnDoneEvent (see page 83)	This event will be triggered when the burning process was completed.
	BurnFileEvent (see page 83)	This event will be triggered for each file in the current burning process.
	CompareFilesForArrangementEvent (see page 83)	This event will will compare file 1 with file 2 and give the status if file1 will be added before file 2. This callback is needed for sorting a disk.
	CreateDirEvent (see page 84)	This event will be triggered after a new directory was created in the current project.
	EraseDoneEvent (see page 84)	This event will be triggered when the deletion process was completed.
	FinalizeEvent (see page 84)	This event will be triggered during the burning process when the finalize process starts. After this event is triggered the Event ProcessEvent (see page 85) will report the progress of the finalize process.
	InfoTextEvent (see page 84)	This event is triggered when the IsoSDK (see page 1) was loaded and an action was performed. This event is useable for logging.

	JobDoneEvent (see page 84)	This event will be triggered when all operations with the device are finished (after burn or erase process) and the IsoSDK (see page 1) is available for further use.
	ProcessEvent (see page 85)	This event will be triggered during a deletion or burn, erase and finalize process. It shows the current progress of the process.
	RemoveFileEvent (see page 85)	This event will be triggered when a file was removed from the project.
	StartVerifyEvent (see page 85)	This event will be triggered when the verify process will start.
	TextEvent (see page 85)	This event will be triggered when you call the GetText (see page 48)() function.
	VerifyDoneEvent (see page 85)	This event will be triggered when the verify process is finished.
	VerifyErrorEvent (see page 86)	This event is triggered if an error occurred during the current verify process.
	VerifyFileEvent (see page 86)	This event will be triggered every time a new file will be verified.
	VerifySectorEvent (see page 86)	This event will be triggered while sector verify (CreateImage (see page 29) / CopyDisc).
	VideoScanDoneEvent (see page 86)	This event will be triggered when the video scanning process is completed.
	VideoScannerEvent (see page 86)	This event will be triggered during a running scanning process. It shows the current progress of the process.

Public Methods

	Name	Description
	Abort (see page 21)	This function interrupts the current job process.
	AddBurnDevice (see page 22)	This function will add a device to the multi burn device list. This is important for multi-device burning.
	AddDir (see page 22)	This function adds a directory to your project.
	AddFile (see page 23)	This function adds a file to your project.
	AudioFileStop (see page 24)	This function stops the internal audio player.
	Burn (see page 24)	This function writes the prepared project to the inserted disc.
	BurnISO (see page 24)	This function writes a defined ISO file to a disc.
new	CheckSignature (see page 25)	Function to check if the disc or image has a signature for encryption or compression.
	ClearAll (see page 25)	This function removes all files and folders from the current project.
	CloseDevice (see page 25)	This is the overview for the CloseDevice method overload.
	CloseSession (see page 26)	This is the overview for the CloseSession method overload.
	ConvertSpeedFromKBPerSec (see page 27)	This function will convert/calculate a given speed in kb/s, like 7200 kb/s, to a readable speed information, like 48.0.
	CopyDisk (see page 28)	This function will copy a disk directly to a selected burner.
	CreateDir (see page 28)	This function creates a new empty directory in the current project.
	CreateImage (see page 29)	This function will copy a disk to a disk image file like ISO or BIN/CUE.
	CreateProject (see page 29)	This is the overview for the CreateProject method overload.
	DeleteProject (see page 30)	This function deletes the current project.
	EjectDevice (see page 30)	This is the overview for the EjectDevice method overload.
	EnableImageDevice (see page 31)	Enable or disable the internal ImageWriter device inside the device list.
	EnableMCNDisabling (see page 32)	This function will enable / disable the media change notification event on windows driven computers. Use this function to work with robotic systems.
	Erase (see page 32)	This function erases the disc in the current device.
	EraseMpegByIndex (see page 33)	This function deletes an added MPEG file from a VCD or SVCD project.
	GetActiveDevicesCount (see page 33)	This function will rescan the internal device list. This is useful to detect new connected devices like USB and FireWire.
	GetASPI (see page 33)	With this function you can get the information about the currently used ASPI layer.

	GetAudioFileSize (see page 34)	Use this function enumerate the final size of the compressed audio file.
new	GetBurnDevices (see page 34)	This function will read the selected burning devices in the IsoSDK (see page 1). This are most time all devices for multidevice burning. You can use AddBurnDevice (see page 22) and RemoveBurnDevice (see page 59) to manage this array.
	GetBurnSpeed (see page 34)	This function will get the writing speed of the device given to the index.
	GetDeviceCapabilities (see page 35)	This is the overview for the GetDeviceCapabilities method overload.
	GetDeviceInformation (see page 36)	This is the overview for the GetDeviceInformation method overload.
	GetDeviceInformationEx (see page 37)	This is the overview for the GetDeviceInformationEx method overload.
	GetDevices (see page 38)	This function will read the available devices of the system according to the selected filter parameter.
	GetExtendedDeviceCapabilities (see page 38)	This is the overview for the GetExtendedDeviceCapabilities method overload.
	GetFileEntry (see page 39)	This function will allow you to get all information about an already added file.
	GetImageFilePath (see page 39)	The function get the path to the image (*.iso or *.bin) file you want to burn using the 'Imagewriter'.
	GetImageSize (see page 39)	This function returns the project size. The GetImageSize function is only available after the execution of the Prepare (see page 58) function.
	GetLastError (see page 40)	This function returns the last error that occurs.
	GetMaxBurnSpeed (see page 40)	This is the overview for the GetMaxBurnSpeed method overload.
	GetMaxReadSpeed (see page 41)	This is the overview for the GetMaxReadSpeed method overload.
	GetMediumInfo (see page 42)	This is the overview for the GetMediumInfo method overload.
	GetMpegCount (see page 43)	This function calculate the number of added files from a VCD or SVCD project.
	GetPlayTime (see page 43)	This function returns the playtime of the given audio file in seconds.
	GetPossibleBurnSpeeds (see page 43)	This is the overview for the GetPossibleBurnSpeeds method overload.
	GetPossibleImageFormats (see page 45)	This function will check what kind of image formats are possible to save from selected disk.
	GetPossibleReadSpeeds (see page 45)	This is the overview for the GetPossibleReadSpeeds method overload.
	GetPrecisePlayTime (see page 46)	This function returns the playtime of the given audio file in milliseconds.
	GetProjectType (see page 47)	This function returns the type of the current project.
	GetReadSpeed (see page 47)	This Function will get the ReadSpead of the device according to the given index.
	GetSessionInfo (see page 47)	This is the overview for the GetSessionInfo method overload.
	GetText (see page 48)	This function returns the text of a TextID, e.g. error code, according to your language setting.
	GetTrackFormatEx (see page 49)	This function returns the type of the disk track of the current medium of the given device.
	GetTrackIndexes (see page 49)	This function reads a list of sub indexes of the give audio track.
	GetTrackInfo (see page 50)	This is the overview for the GetTrackInfo method overload.
	GetTrackISRC (see page 51)	This function reads the ISRC code from the give audio track.
	GrabAudioTrack (see page 51)	The function will grab/save a Audio Track to a file. Tagging and different Encoding types are available.
	Initialize (see page 52)	This function initializes the IsoSDK (see page 1). The function must be called before you can use other functions.
	IsDeviceReady (see page 52)	This is the overview for the IsDeviceReady method overload.

	IsValidVideoTsFolder (see page 53)	This function will check if a given path contains a valid VideoDVD structure with IFO, BUP and VOB Files.
	LoadBassPlugin (see page 54)	This function will load a given Bass plugin to the IsoSDK (see page 1). Bass.dll have to be present. http://www.un4seen.com/ **NOTE:** The bass.dll and plugins are available as 32 and 64 bit.
	LockMedium (see page 54)	This is the overview for the LockMedium method overload.
	OpenDiskSession (see page 55)	This is the overview for the OpenDiskSession method overload.
	PlayAudioFile (see page 57)	Use this function to play the given file to the default output device.
	PlayAudioTrack (see page 57)	This function will play a audio track on the current medium or disk image file.
	Prepare (see page 58)	This function prepares the data to be written. In this step the IsoSDK (see page 1) creates the file system and all tables.
	ReadCDText (see page 58)	This function will create a handle to the CD-Text information of the audio disk according to the selected device.
	ReadSectors (see page 58)	This function will read a given sector into a buffer.
	RemoveBurnDevice (see page 59)	This function will remove a device from the multi burn device list. This is important for multi device burning.
	RemoveDir (see page 59)	This function renames/moves a directory in the project.
	RemoveFile (see page 60)	This function removes a file from the project.
	RenameDir (see page 60)	This function renames/moves file in the project.
	RenameFile (see page 61)	This function renames/moves directory in the project.
	RescanDevices (see page 61)	This function will rescan the internal device list. This is useful to detect new connected devices like USB and FireWire.
	SaveLogToFile (see page 62)	This function will save the automatically generated burning log data to a file.
	SaveTrackToFile (see page 62)	This function saves a selected track to a file.
	SetASPI (see page 63)	If you want to use the internal ASPI layer after the initialization you use this function to switch from the external to the internal ASPI layer et vice versa .
	SetAudioFileProperty (see page 63)	The function sets the audio properties for the specified audio file in the project or for the disk. Supported projects: Audio and Mixed Mode.
	SetBurnSpeed (see page 64)	This function sets the writing speed of the current device.
	SetFileAttr (see page 64)	This function set the attributes to a file.
	SetFileTimes (see page 65)	This function will set the custom dates to a file.
	SetFileUserParam (see page 65)	This is a data value you can set yourself according to your needs. You can pass a integer or structure to the file that stores extra data.
	SetImageFilePath (see page 66)	The function sets the path to the image (*.iso) file you want to burn using the 'Imagewriter'.
	SetLanguage (see page 66)	This function sets the language of the IsoSDK (see page 1). The following resources are effected: error codes, message codes and GUI codes.
	SetRawStructure (see page 67)	The function sets the structure of the disc and the format of the input data for a RAW project.
	SetReadSpeed (see page 67)	This function sets the reading speed of the device according to the given index.
	SetRegionalCode (see page 68)	This is the overview for the SetRegionalCode method overload.
	SetVCDKeyHandler (see page 69)	This function sets which item will be shown next when particular key is pressed.
	SetVCDTimeOutHandler (see page 70)	This function sets transition timeout between items.
	StopMpegAction (see page 70)	This function kills the current scanning process of an added video file to a VCD or SVCD project.

	TagsFromNetworkDialog (see page 71)	This function will create a handle to the NetworkTags (see page 122) information of the audio disk according to the selected device.

Public Properties

	Name	Description
R	BootOptions (see page 72)	This property is a pointer to the IsoSDK::BootOptions (see page 10) Class to set/get BootDisk properties / settings.
	BurnDevice (see page 72)	This property sets and gets the current burn device. This drive will be used for the burn process.
	BurnSpeed (see page 72)	This property set and get the writing speed of the current device.
R	CompressEncryptOptions (see page 73)	This property is a pointer to the IsoSDK::CompressEncryptOptions (see page 90) Class to set/get properties / settings for encryption and compressing.
R	DVDVideoOptions (see page 73)	This property is a pointer to the IsoSDK::DVDVideoOptions (see page 98) Class to set/get extended VideoDVD properties / settings.
R	FileDateTimeEx (see page 73)	This is FileDateTimeEx, a member of class Burner.
R	FirstSegmentIndex (see page 73)	Represents the first segment to play on a VCDKeyHandler. Default starts at 1000.
R	FirstTrackIndex (see page 73)	Represents the first track to play on a VCDKeyHandler. Default starts at 0.
	ImageDeviceEnabled (see page 74)	This property set and get the state of the internal ImageDevice switch.
R	ISOExOptions (see page 74)	This property is a pointer to the IsoSDK::ISOExOptions (see page 114) Class to set/get ISO Ex options properties / settings.
	Language (see page 74)	This property set and get the language of the IsoSDK (see page 1). The following resources are effected: error codes, message codes and GUI codes.
R	Options (see page 74)	This property is a pointer to the IsoSDK::GeneralOptions (see page 107) Class to set/get general options properties / settings.
	ReadDevice (see page 74)	This property sets and gets the current read device. This drive will be used for the read process.
	ReadSpeed (see page 75)	This property set and get the reading speed of the current device.
	TmpPath (see page 75)	This property sets and gets the reading speed of the current device.
R	UDFOptions (see page 75)	This property is a pointer to the IsoSDK::UDFOptions (see page 128) Class to set/get UDFOptions properties / settings.
R	VCDInfiniteTimeout (see page 75)	This property represents the timeout value used in SetVCDTimeOutHandler (see page 70) method.
R	VCDKey0 (see page 75)	User interaction (Button). disabled with $FFFD. Else it will play the selected item in the menu selection.
R	VCDKeyDefault (see page 76)	User interaction (Button). This item stops current playback or wait state and returns to the default list item or given item in play list.
R	VCDKeyNext (see page 76)	User interaction (Button). This item stops current playback or wait state and plays the next item in the play list.
R	VCDKeyPrevious (see page 76)	User interaction (Button). This item stops current playback or wait state and plays the previous item in the play list.
R	VCDKeyReturn (see page 76)	User interaction (Button). This item stops current playback or wait state and returns to menu or given item in play list.
	Verify (see page 76)	Use this property to get and set the after burn verification state.
	WriteCDTextInUnicode (see page 77)	Get/Set the property to write CD-Text with Unicode or Multibyte.

1.1.1.6.2 Burner Methods

The methods of the Burner class are listed here.

Public Methods

	Name	Description
	Abort (see page 21)	This function interrupts the current job process.
	AddBurnDevice (see page 22)	This function will add a device to the multi burn device list. This is important for multi-device burning.
	AddDir (see page 22)	This function adds a directory to your project.
	AddFile (see page 23)	This function adds a file to your project.
	AudioFileStop (see page 24)	This function stops the internal audio player.
	Burn (see page 24)	This function writes the prepared project to the inserted disc.
	BurnISO (see page 24)	This function writes a defined ISO file to a disc.
new	CheckSignature (see page 25)	Function to check if the disc or image has a signature for encryption or compression.
	ClearAll (see page 25)	This function removes all files and folders from the current project.
	CloseDevice (see page 25)	This is the overview for the CloseDevice method overload.
	CloseSession (see page 26)	This is the overview for the CloseSession method overload.
	ConvertSpeedFromKBPerSec (see page 27)	This function will convert/calculate a given speed in kb/s, like 7200 kb/s, to a readable speed information, like 48.0.
	CopyDisk (see page 28)	This function will copy a disk directly to a selected burner.
	CreateDir (see page 28)	This function creates a new empty directory in the current project.
	CreateImage (see page 29)	This function will copy a disk to a disk image file like ISO or BIN/CUE.
	CreateProject (see page 29)	This is the overview for the CreateProject method overload.
	DeleteProject (see page 30)	This function deletes the current project.
	EjectDevice (see page 30)	This is the overview for the EjectDevice method overload.
	EnableImageDevice (see page 31)	Enable or disable the internal ImageWriter device inside the device list.
	EnableMCNDisabling (see page 32)	This function will enable / disable the media change notification event on windows driven computers. Use this function to work with robotic systems.
	Erase (see page 32)	This function erases the disc in the current device.
	EraseMpegByIndex (see page 33)	This function deletes an added MPEG file from a VCD or SVCD project.
	GetActiveDevicesCount (see page 33)	This function will rescan the internal device list. This is useful to detect new connected devices like USB and FireWire.
	GetASPI (see page 33)	With this function you can get the information about the currently used ASPI layer.
	GetAudioFileSize (see page 34)	Use this function enumerate the final size of the compressed audio file.
new	GetBurnDevices (see page 34)	This function will read the selected burning devices in the IsoSDK (see page 1). This are most time all devices for multidevice burning. You can use AddBurnDevice (see page 22) and RemoveBurnDevice (see page 59) to manage this array.
	GetBurnSpeed (see page 34)	This function will get the writing speed of the device given to the index.
	GetDeviceCapabilities (see page 35)	This is the overview for the GetDeviceCapabilities method overload.
	GetDeviceInformation (see page 36)	This is the overview for the GetDeviceInformation method overload.
	GetDeviceInformationEx (see page 37)	This is the overview for the GetDeviceInformationEx method overload.
	GetDevices (see page 38)	This function will read the available devices of the system according to the selected filter parameter.
	GetExtendedDeviceCapabilities (see page 38)	This is the overview for the GetExtendedDeviceCapabilities method overload.
	GetFileEntry (see page 39)	This function will allow you to get all information about an already added file.
	GetImageFilePath (see page 39)	The function get the path to the image (*.iso or *.bin) file you want to burn using the 'Imagewriter'.

	GetImageSize (see page 39)	This function returns the project size. The GetImageSize function is only available after the execution of the Prepare (see page 58) function.
	GetLastError (see page 40)	This function returns the last error that occurs.
	GetMaxBurnSpeed (see page 40)	This is the overview for the GetMaxBurnSpeed method overload.
	GetMaxReadSpeed (see page 41)	This is the overview for the GetMaxReadSpeed method overload.
	GetMediumInfo (see page 42)	This is the overview for the GetMediumInfo method overload.
	GetMpegCount (see page 43)	This function calculate the number of added files from a VCD or SVCD project.
	GetPlayTime (see page 43)	This function returns the playtime of the given audio file in seconds.
	GetPossibleBurnSpeeds (see page 43)	This is the overview for the GetPossibleBurnSpeeds method overload.
	GetPossibleImageFormats (see page 45)	This function will check what kind of image formats are possible to save from selected disk.
	GetPossibleReadSpeeds (see page 45)	This is the overview for the GetPossibleReadSpeeds method overload.
	GetPrecisePlayTime (see page 46)	This function returns the playtime of the given audio file in milliseconds.
	GetProjectType (see page 47)	This function returns the type of the current project.
	GetReadSpeed (see page 47)	This Function will get the ReadSpead of the device according to the given index.
	GetSessionInfo (see page 47)	This is the overview for the GetSessionInfo method overload.
	GetText (see page 48)	This function returns the text of a TextID, e.g. error code, according to your language setting.
	GetTrackFormatEx (see page 49)	This function returns the type of the disk track of the current medium of the given device.
	GetTrackIndexes (see page 49)	This function reads a list of sub indexes of the give audio track.
	GetTrackInfo (see page 50)	This is the overview for the GetTrackInfo method overload.
	GetTrackISRC (see page 51)	This function reads the ISRC code from the give audio track.
	GrabAudioTrack (see page 51)	The function will grab/save a Audio Track to a file. Tagging and different Encoding types are available.
	Initialize (see page 52)	This function initializes the IsoSDK (see page 1). The function must be called before you can use other functions.
	IsDeviceReady (see page 52)	This is the overview for the IsDeviceReady method overload.
	IsValidVideoTsFolder (see page 53)	This function will check if a given path contains a valid VideoDVD structure with IFO, BUP and VOB Files.
	LoadBassPlugin (see page 54)	This function will load a given Bass plugin to the IsoSDK (see page 1). Bass.dll have to be present. http://www.un4seen.com/ **NOTE:** The bass.dll and plugins are available as 32 and 64 bit.
	LockMedium (see page 54)	This is the overview for the LockMedium method overload.
	OpenDiskSession (see page 55)	This is the overview for the OpenDiskSession method overload.
	PlayAudioFile (see page 57)	Use this function to play the given file to the default output device.
	PlayAudioTrack (see page 57)	This function will play a audio track on the current medium or disk image file.
	Prepare (see page 58)	This function prepares the data to be written. In this step the IsoSDK (see page 1) creates the file system and all tables.
	ReadCDText (see page 58)	This function will create a handle to the CD-Text information of the audio disk according to the selected device.
	ReadSectors (see page 58)	This function will read a given sector into a buffer.
	RemoveBurnDevice (see page 59)	This function will remove a device from the multi burn device list. This is important for multi device burning.
	RemoveDir (see page 59)	This function renames/moves a directory in the project.

	RemoveFile (see page 60)	This function removes a file from the project.
	RenameDir (see page 60)	This function renames/moves file in the project.
	RenameFile (see page 61)	This function renames/moves directory in the project.
	RescanDevices (see page 61)	This function will rescan the internal device list. This is useful to detect new connected devices like USB and FireWire.
	SaveLogToFile (see page 62)	This function will save the automatically generated burning log data to a file.
	SaveTrackToFile (see page 62)	This function saves a selected track to a file.
	SetASPI (see page 63)	If you want to use the internal ASPI layer after the initialization you use this function to switch from the external to the internal ASPI layer et vice versa .
	SetAudioFileProperty (see page 63)	The function sets the audio properties for the specified audio file in the project or for the disk. Supported projects: Audio and Mixed Mode.
	SetBurnSpeed (see page 64)	This function sets the writing speed of the current device.
	SetFileAttr (see page 64)	This function set the attributes to a file.
	SetFileTimes (see page 65)	This function will set the custom dates to a file.
	SetFileUserParam (see page 65)	This is a data value you can set yourself according to your needs. You can pass a integer or structure to the file that stores extra data.
	SetImageFilePath (see page 66)	The function sets the path to the image (*.iso) file you want to burn using the 'Imagewriter'.
	SetLanguage (see page 66)	This function sets the language of the IsoSDK (see page 1). The following resources are effected: error codes, message codes and GUI codes.
	SetRawStructure (see page 67)	The function sets the structure of the disc and the format of the input data for a RAW project.
	SetReadSpeed (see page 67)	This function sets the reading speed of the device according to the given index.
	SetRegionalCode (see page 68)	This is the overview for the SetRegionalCode method overload.
	SetVCDKeyHandler (see page 69)	This function sets which item will be shown next when particular key is pressed.
	SetVCDTimeOutHandler (see page 70)	This function sets transition timeout between items.
	StopMpegAction (see page 70)	This function kills the current scanning process of an added video file to a VCD or SVCD project.
	TagsFromNetworkDialog (see page 71)	This function will create a handle to the NetworkTags (see page 122) information of the audio disk according to the selected device.

1.1.1.6.2.1 Burner::Abort Method

C++

```
bool Abort();
```

C#

```
public bool Abort();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function interrupts the current job process.

Notes

Depending on the system or device, the abort process may take some time to be completed.

1.1.1.6.2.2 Burner::AddBurnDevice Method

C++

```
bool AddBurnDevice(
    String ^ Device
);
```

C#

```
public bool AddBurnDevice(
    ref String Device
);
```

Parameters

Parameters	Description
String ^ Device	The device/drive string of the device to be added.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will add a device to the multi burn device list. This is important for multi-device burning.

Notes

Only the first letter or number of the device string will be used.

1.1.1.6.2.3 Burner::AddDir Method

C++

```
bool AddDir(
    String ^ SourceDirPath,
    String ^ DestinationPath,
    String ^ FileSpecification,
    FileAttributes Attributes,
    bool Recursive,
    SavePathOption SavePath
);
```

C#

```
public bool AddDir(
    ref String SourceDirPath,
    ref String DestinationPath,
    ref String FileSpecification,
    FileAttributes Attributes,
    bool Recursive,
    SavePathOption SavePath
);
```

Parameters

Parameters	Description
String ^ SourceDirPath	The folder path you want to add to the project.
String ^ DestinationPath	Path of the destination folder of the project.
String ^ FileSpecification	Specifies the file filter to be applied to the folder. Examples: „*.*“ or „*.doc“ or „MyPrefix_*.*“
FileAttributes Attributes	The file attributes according to the enum FileAttributes (see page 184).
bool Recursive	This argument states whether the SDK will add the underlaying file. Recursive = true (default). If you only want to add the named folder you have to set it to false.

SavePathOption SavePath	This argument states if and how the super ordinate directory is added as a folder to the project according to the enum SavePathOption (see page 199).

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function adds a directory to your project.

Notes

This function is dedicated to small and predefined directories like those for VideoDVD or Blu-Ray Video. The common way to add files and folders is to loop through the path and then use AddFile (see page 23) and CreateDir (see page 28). This will give you the possibility to use also SetFileTimes (see page 65) and SetFileAttr (see page 64) and RenameFile (see page 61) functions for each file.

1.1.1.6.2.4 Burner::AddFile Method

C++

```
bool AddFile(
    String ^ SourceFilePath,
    String ^ DestinationPath,
    String ^ FileName,
    bool VideoFile,
    SavePathOption SavePath
);
```

C#

```
public bool AddFile(
    ref String SourceFilePath,
    ref String DestinationPath,
    ref String FileName,
    bool VideoFile,
    SavePathOption SavePath
);
```

Parameters

Parameters	Description
String ^ SourceFilePath	A string with the path of the folder containing the file you want to add to the project. For Cue projects please use the CUE file not the BIN file.
String ^ DestinationPath	A string with the path of the destination folder of the project.
String ^ FileName	A string with the name of the file that you want to add to the project. With audio files for AudioCD or MixedModeCD project leave this value "".
bool VideoFile	A bool value that indicate whether a video file or not.
SavePathOption SavePath	This argument states if and how the super ordinate directory is added as a folder to the project according to the enum SavePathOption (see page 199).

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function adds a file to your project.

Remarks

The value DestinationPath is basing on the rootDirectory The rootDirectory is always "\\" expected MixedMode project. Here the root for audio is "\\audio" and for data files "\\data".

This function returns an error if the file exceed the 2GB files size limit for ISO file system.

1.1.1.6.2.5 Burner::AudioFileStop Method

C++

```
bool AudioFileStop();
```

C#

```
public bool AudioFileStop();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function stops the internal audio player.

1.1.1.6.2.6 Burner::Burn Method

C++

```
bool Burn();
```

C#

```
public bool Burn();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function writes the prepared project to the inserted disc.

Notes

Before you can use this method, the project must have been prepared for burning with the help of the Prepare (see page 58) function.

A disc or disc image should have at least 705,600 bytes.

1.1.1.6.2.7 Burner::BurnISO Method

C++

```
bool BurnISO(
    String ^ FilePath,
    BurnIsoOptions ^ Options
);
```

C#

```
public bool BurnISO(
    ref String FilePath,
    ref BurnIsoOptions Options
);
```

Parameters

Parameters	Description
String ^ FilePath	The path and name of the ISO file to be written.

BurnIsoOptions ^ Options	A structure BurnIsoOptions (see page 145).

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function writes a defined ISO file to a disc.

Notes

The IsoSDK (see page 1) was designed to write ISO compatible files. If the selected ISO file was not built with the IsoSDK (see page 1), problems might occur.

1.1.1.6.2.8 Burner::CheckSignature Method new

C++

```
int CheckSignature();
```

C#

```
public int CheckSignature();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

Function to check if the disc or image has a signature for encryption or compression.

1.1.1.6.2.9 Burner::ClearAll Method

C++

```
bool ClearAll();
```

C#

```
public bool ClearAll();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function removes all files and folders from the current project.

1.1.1.6.2.10 CloseDevice Method

This is the overview for the CloseDevice method overload.

Overload List

	Name	Description
	Burner::CloseDevice () (see page 26)	This function closes the current device (tray).
	Burner::CloseDevice (int) (see page 26)	This function closes the device according to the given index.

1.1.1.6.2.10.1 Burner::CloseDevice Method ()

C++

```
bool CloseDevice();
```

C#

```
public bool CloseDevice();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function closes the current device (tray).

1.1.1.6.2.10.2 Burner::CloseDevice Method (int)

C++

```
bool CloseDevice(
    int Index
);
```

C#

```
public bool CloseDevice(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function closes the device according to the given index.

1.1.1.6.2.11 CloseSession Method

This is the overview for the CloseSession method overload.

Overload List

	Name	Description
	Burner::CloseSession () (see page 26)	This function closes the session of the current device.
	Burner::CloseSession (int) (see page 27)	This function closes the session of the device according to the given index.

1.1.1.6.2.11.1 Burner::CloseSession Method ()

C++

```
bool CloseSession();
```

C#

```
public bool CloseSession();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function closes the session of the current device.

1.1.1.6.2.11.2 Burner::CloseSession Method (int)

C++

```
bool CloseSession(
    int Index
);
```

C#

```
public bool CloseSession(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function closes the session of the device according to the given index.

1.1.1.6.2.12 Burner::ConvertSpeedFromKBPerSec Method

C++

```
bool ConvertSpeedFromKBPerSec(
    int32 SpeedInKBPerSec,
    float * pfConvertedSpeed
);
```

C#

```
public bool ConvertSpeedFromKBPerSec(
    int32 SpeedInKBPerSec,
    ref float pfConvertedSpeed
);
```

Parameters

Parameters	Description
int32 SpeedInKBPerSec	The speed in kb/s you need to convert to readable speed. Like 7200 to get 48.0.
float * pfConvertedSpeed	Pointer to the readable speed information like 48.0.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will convert/calculate a given speed in kb/s, like 7200 kb/s, to a readable speed information, like 48.0.

Notes

You can use this function for internal calculations. this function represents CD write/read speeds. This is not valid for DVD or Blu-ray.

1.1.1.6.2.13 Burner::CopyDisk Method

C++

```
bool CopyDisk(
    DiskCopyOptions ^ cDiskCopyOptions
);
```

C#

```
public bool CopyDisk(
    ref DiskCopyOptions cDiskCopyOptions
);
```

Parameters

Parameters	Description
DiskCopyOptions ^ cDiskCopyOptions	A structure DiskCopyOptions (see page 154).

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will copy a disk directly to a selected burner.

Notes

Make sure that you set the properties ReadDevice (see page 74) and BurnDevice (see page 72) before you call this function.

1.1.1.6.2.14 Burner::CreateDir Method

C++

```
bool CreateDir(
    String ^ Dir,
    String ^ DestinationPath
);
```

C#

```
public bool CreateDir(
    ref String Dir,
    ref String DestinationPath
);
```

Parameters

Parameters	Description
String ^ Dir	This is the name of the new directory. Only one plane is allowed. (e. g. „new“, but not „new\new“).
String ^ DestinationPath	In which destination path in your project do you want to create the new directory.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function creates a new empty directory in the current project.

1.1.1.6.2.15 Burner::CreateImage Method

C++

```
bool CreateImage(
    CreateImageParams ^ Params,
    ImageTask TaskType
);
```

C#

```
public bool CreateImage(
    ref CreateImageParams Params,
    ImageTask TaskType
);
```

Parameters

Parameters	Description
CreateImageParams ^ Params	A structure CreateImageParams (see page 152).
ImageTask TaskType	The type of information according to a ImageTask (see page 187) enumeration.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will copy a disk to a disk image file like ISO or BIN/CUE.

1.1.1.6.2.16 CreateProject Method

This is the overview for the CreateProject method overload.

Overload List

	Name	Description
	Burner::CreateProject (ProjectType) (see page 29)	This function creates a new project. The current project will be deleted. This function will create a project without any session.
	Burner::CreateProject (ProjectType, int) (see page 30)	This function creates a new project. The current project will be deleted.

1.1.1.6.2.16.1 Burner::CreateProject Method (ProjectType)

C++

```
bool CreateProject(
    ProjectType Type
);
```

C#

```
public bool CreateProject(
    ProjectType Type
);
```

Parameters

Parameters	Description
ProjectType Type	The project type according to a ProjectType (see page 195) enumeration.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function creates a new project. The current project will be deleted. This function will create a project without any session.

1.1.1.6.2.16.2 Burner::CreateProject Method (ProjectType, int)

C++

```
bool CreateProject(
    ProjectType Type,
    int SessionToContinue
);
```

C#

```
public bool CreateProject(
    ProjectType Type,
    int SessionToContinue
);
```

Parameters

Parameters	Description
ProjectType Type	The project type according to a ProjectType (see page 195) enumeration.
int SessionToContinue	The session that will be imported to the new session. Use -1 to import no session or 0 to import the last session. This parameter is only valid for data projects and will be ignored in other projects.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function creates a new project. The current project will be deleted.

1.1.1.6.2.17 Burner::DeleteProject Method

C++

```
bool DeleteProject();
```

C#

```
public bool DeleteProject();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function deletes the current project.

Notes

Use this function only when terminating the program or if you want to create a new project.

1.1.1.6.2.18 EjectDevice Method

This is the overview for the EjectDevice method overload.

Overload List

	Name	Description
	Burner::EjectDevice () (see page 31)	This function opens the current device (tray).
	Burner::EjectDevice (int) (see page 31)	This function opens the tray according to the given device index.

1.1.1.6.2.18.1 Burner::EjectDevice Method ()

C++

```
bool EjectDevice();
```

C#

```
public bool EjectDevice();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function opens the current device (tray).

1.1.1.6.2.18.2 Burner::EjectDevice Method (int)

C++

```
bool EjectDevice(
    int Index
);
```

C#

```
public bool EjectDevice(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function opens the tray according to the given device index.

1.1.1.6.2.19 Burner::EnableImageDevice Method

C++

```
bool EnableImageDevice(
    bool Enable
);
```

C#

```
public bool EnableImageDevice(
    bool Enable
);
```

Parameters

Parameters	Description
bool Enable	This value will enable or disable the internal image writer. The image writer is used to create ISO disk images from your projects.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

Enable or disable the internal ImageWriter device inside the device list.

1.1.1.6.2.20 Burner::EnableMCNDisabling Method

C++

```
bool EnableMCNDisabling(
    bool Enable
);
```

C#

```
public bool EnableMCNDisabling(
    bool Enable
);
```

Parameters

Parameters	Description
bool Enable	A bool value to disable (false) or enable (true) the media change notification.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will enable / disable the media change notification event on windows driven computers. Use this function to work with robotic systems.

1.1.1.6.2.21 Burner::Erase Method

C++

```
bool Erase(
    bool Fast,
    bool Eject
);
```

C#

```
public bool Erase(
    bool Fast,
    bool Eject
);
```

Parameters

Parameters	Description
bool Fast	This parameter states whether the medium is erased using the fast or complete method. Pass **true** to fast erase the medium, and **false** to erase the medium completely (this method lasts much longer).

bool Eject	This parameter states whether the medium will be ejected after the erase process is complete.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function erases the disc in the current device.

Notes

In most cases, it is recommended to fast erase the disc.

Important: On Windows OS it is recommended to eject the medium after erase is complete. Else Windows have problems to recognize that the disc is empty / changed.

1.1.1.6.2.22 Burner::EraseMpegByIndex Method

C++

```
bool EraseMpegByIndex(
    int Index
);
```

C#

```
public bool EraseMpegByIndex(
    int Index
);
```

Parameters

Parameters	Description
int Index	This value holds the position in the current file list. It starts from 0 to x.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function deletes an added MPEG file from a VCD or SVCD project.

1.1.1.6.2.23 Burner::GetActiveDevicesCount Method

C++

```
int GetActiveDevicesCount();
```

C#

```
public int GetActiveDevicesCount();
```

Returns

This function will return the amount of devices found my the IsoSDK (see page 1).

Description

This function will rescan the internal device list. This is useful to detect new connected devices like USB and FireWire.

1.1.1.6.2.24 Burner::GetASPI Method

C++

```
ASPIInterface GetASPI();
```

C#

```
public ASPIInterface GetASPI();
```

Returns

An value according to the ASPIInterface (see page 141) enumeration.

Description

With this function you can get the information about the currently used ASPI layer.

1.1.1.6.2.25 Burner::GetAudioFileSize Method

C++

```
double GetAudioFileSize(
    String ^ FileName
);
```

C#

```
public double GetAudioFileSize(
    ref String FileName
);
```

Parameters

Parameters	Description
String ^ FileName	The file to calculate the file size. File name with full path.

Returns

This function returns a double value with the final decoded file size on the disk.

Description

Use this function enumerate the final size of the compressed audio file.

1.1.1.6.2.26 Burner::GetBurnDevices Method new

C++

```
array<String ^> ^ GetBurnDevices();
```

C#

```
public array<String ^> GetBurnDevices();
```

Returns

The function will return a string array with the devices available.

Description

This function will read the selected burning devices in the IsoSDK (see page 1). This are most time all devices for multidevice burning. You can use AddBurnDevice (see page 22) and RemoveBurnDevice (see page 59) to manage this array.

1.1.1.6.2.27 Burner::GetBurnSpeed Method

C++

```
int GetBurnSpeed(
    int Index
);
```

C#

```
public int GetBurnSpeed(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The burn speed of the selected device.

Description

This function will get the writing speed of the device given to the index.

1.1.1.6.2.28 GetDeviceCapabilities Method

This is the overview for the GetDeviceCapabilities method overload.

Overload List

	Name	Description
	Burner::GetDeviceCapabilities () (see page 35)	This function will get the device capabilities of the current device.
	Burner::GetDeviceCapabilities (int) (see page 35)	This function will get the device capabilities of the device according to the given index.

1.1.1.6.2.28.1 Burner::GetDeviceCapabilities Method ()

C++

```
Capabilities GetDeviceCapabilities();
```

C#

```
public Capabilities GetDeviceCapabilities();
```

Returns

This function returns a value according to the Capabilities (see page 146) enumeration.

Description

This function will get the device capabilities of the current device.

Example

You can read the possible values with a AND query:

```
Capabilities cCap = m_burner.GetDeviceCapabilities();
checkBoxReadCDR.Checked = ((long)(cCap & Capabilities.ReadCdR) == 0) ? false : true;
```

1.1.1.6.2.28.2 Burner::GetDeviceCapabilities Method (int)

C++

```
Capabilities GetDeviceCapabilities(
    int Index
);
```

C#

```
public Capabilities GetDeviceCapabilities(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This function returns a value according to the Capabilities (see page 146) enumeration.

Description

This function will get the device capabilities of the device according to the given index.

Example

You can read the possible values with a AND query:

```
Capabilities cCap = m_burner.GetDeviceCapabilities(2);
checkBoxReadCDR.Checked = ((long)(cCap & Capabilities.ReadCdR) == 0) ? false : true;
```

1.1.1.6.2.29 GetDeviceInformation Method

This is the overview for the GetDeviceInformation method overload.

Overload List

	Name	Description
	Burner::GetDeviceInformation () (see page 36)	This function reads the device information of the current device.
	Burner::GetDeviceInformation (int) (see page 36)	This function reads the device information of the device given by the selected index.

1.1.1.6.2.29.1 Burner::GetDeviceInformation Method ()

C++

```
DeviceInformation ^ GetDeviceInformation();
```

C#

```
public DeviceInformation GetDeviceInformation();
```

Returns

This functions returns a reference to a filled structure DeviceInformation (see page 153).

Description

This function reads the device information of the current device.

1.1.1.6.2.29.2 Burner::GetDeviceInformation Method (int)

C++

```
DeviceInformation ^ GetDeviceInformation(
    int Index
);
```

C#

```
public DeviceInformation GetDeviceInformation(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This functions returns a reference to a filled structure DeviceInformation (see page 153).

Description

This function reads the device information of the device given by the selected index.

1.1.1.6.2.30 GetDeviceInformationEx Method

This is the overview for the GetDeviceInformationEx method overload.

Overload List

	Name	Description
	Burner::GetDeviceInformationEx () (see page 37)	This function reads the extended device information of the current device.
	Burner::GetDeviceInformationEx (int) (see page 37)	This function reads the extended device information of the device given by the selected index.

1.1.1.6.2.30.1 Burner::GetDeviceInformationEx Method ()

C++

```
ExtendedDeviceInformation ^ GetDeviceInformationEx();
```

C#

```
public ExtendedDeviceInformation GetDeviceInformationEx();
```

Returns

This functions returns a reference to a filled structure ExtendedDeviceInformation (see page 181).

Description

This function reads the extended device information of the current device.

1.1.1.6.2.30.2 Burner::GetDeviceInformationEx Method (int)

C++

```
ExtendedDeviceInformation ^ GetDeviceInformationEx(
    int Index
);
```

C#

```
public ExtendedDeviceInformation GetDeviceInformationEx(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This functions returns a reference to a filled structure ExtendedDeviceInformation (see page 181).

Description

This function reads the extended device information of the device given by the selected index.

1.1.1.6.2.31 Burner::GetDevices Method

C++

```
array<String ^> ^ GetDevices(
    [In] bool BurningDevicesOnly
);
```

C#

```
public array<String ^> GetDevices(
    [In] bool BurningDevicesOnly
);
```

Parameters

Parameters	Description
[In] bool BurningDevicesOnly	A bool value to enable/disable the filter. true will scan only burn devices and false all devices.

Returns

The function will return a string array with the devices available.

Description

This function will read the available devices of the system according to the selected filter parameter.

1.1.1.6.2.32 GetExtendedDeviceCapabilities Method

This is the overview for the GetExtendedDeviceCapabilities method overload.

Overload List

	Name	Description
	Burner::GetExtendedDeviceCapabilities () (see page 38)	This function reads the extended device capabilities of the current device.
	Burner::GetExtendedDeviceCapabilities (int) (see page 38)	This function reads the extended device capabilities of the device given by the selected index.

1.1.1.6.2.32.1 Burner::GetExtendedDeviceCapabilities Method ()

C++

```
ExtendedDeviceCapabilities ^ GetExtendedDeviceCapabilities();
```

C#

```
public ExtendedDeviceCapabilities GetExtendedDeviceCapabilities();
```

Returns

This functions returns a reference to a filled structure ExtendedDeviceCapabilities (see page 100).

Description

This function reads the extended device capabilities of the current device.

1.1.1.6.2.32.2 Burner::GetExtendedDeviceCapabilities Method (int)

C++

```
ExtendedDeviceCapabilities ^ GetExtendedDeviceCapabilities(
    int Index
);
```

C#

```
public ExtendedDeviceCapabilities GetExtendedDeviceCapabilities(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This functions returns a reference to a filled structure ExtendedDeviceCapabilities (see page 100).

Description

This function reads the extended device capabilities of the device given by the selected index.

1.1.1.6.2.33 Burner::GetFileEntry Method

C++

```
FileEntry ^ GetFileEntry(
    String ^ filePath
);
```

C#

```
public FileEntry GetFileEntry(
    ref String filePath
);
```

Parameters

Parameters	Description
String ^ filePath	File name + full path of the file to get the information from.

Returns

A pointer to a filled structure FileEntry (see page 185).

Description

This function will allow you to get all information about an already added file.

1.1.1.6.2.34 Burner::GetImageFilePath Method

C++

```
String ^ GetImageFilePath();
```

C#

```
public String GetImageFilePath();
```

Returns

The function will return a string containing the ImageFilePath.

Description

The function get the path to the image (*.iso or *.bin) file you want to burn using the 'Imagewriter'.

1.1.1.6.2.35 Burner::GetImageSize Method

C++

```
double GetImageSize();
```

C#

```
public double GetImageSize();
```

Returns

This function returns the project size in bytes. The return parameter is a Double value.

Description

This function returns the project size. The GetImageSize function is only available after the execution of the Prepare (see page 58) function.

Notes

The functions Prepare (see page 58)() must have been called before this function call.

1.1.1.6.2.36 Burner::GetLastError Method

C++

```
ErrorCode GetLastError();
```

C#

```
public ErrorCode GetLastError();
```

Returns

An value according to the ErrorCode (see page 155) enumeration.

Description

This function returns the last error that occurs.

1.1.1.6.2.37 GetMaxBurnSpeed Method

This is the overview for the GetMaxBurnSpeed method overload.

Overload List

	Name	Description
	Burner::GetMaxBurnSpeed () (see page 40)	This function returns the max. writing speed of the current device.
	Burner::GetMaxBurnSpeed (int) (see page 41)	This function returns the max. writing speed of the device given by the selected index.

1.1.1.6.2.37.1 Burner::GetMaxBurnSpeed Method ()

C++

```
int GetMaxBurnSpeed();
```

C#

```
public int GetMaxBurnSpeed();
```

Returns

The IsoSDK (see page 1) returns only the highest value. You may want to use a different speed setting. Valid speed settings are usually calculated this way: 1x, 2x as default, 4 and multiples of 4 till the max. burning speed is reached.

Description

This function returns the max. writing speed of the current device.

1.1.1.6.2.37.2 Burner::GetMaxBurnSpeed Method (int)

C++

```
int GetMaxBurnSpeed(
    int Index
);
```

C#

```
public int GetMaxBurnSpeed(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The IsoSDK (see page 1) returns only the highest value. You may want to use a different speed setting. Valid speed settings are usually calculated this way: 1x, 2x as default, 4 and multiples of 4 till the max. burning speed is reached.

Description

This function returns the max. writing speed of the device given by the selected index.

1.1.1.6.2.38 GetMaxReadSpeed Method

This is the overview for the GetMaxReadSpeed method overload.

Overload List

	Name	Description
	Burner::GetMaxReadSpeed () (see page 41)	This function reads the max. reading speed of the current device.
	Burner::GetMaxReadSpeed (int) (see page 41)	This function reads the max. reading speed of the device given by the selected index.

1.1.1.6.2.38.1 Burner::GetMaxReadSpeed Method ()

C++

```
int GetMaxReadSpeed();
```

C#

```
public int GetMaxReadSpeed();
```

Returns

The IsoSDK (see page 1) returns only the highest value. You may want to use a different speed setting. Valid speed settings are usually calculated this way: 1x, 2x as default, 4 and multiples of 4 till the max. burning speed is reached.

Description

This function reads the max. reading speed of the current device.

1.1.1.6.2.38.2 Burner::GetMaxReadSpeed Method (int)

C++

```
int GetMaxReadSpeed(
    int Index
);
```

C#

```
public int GetMaxReadSpeed(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The IsoSDK (see page 1) returns only the highest value. You may want to use a different speed setting. Valid speed settings are usually calculated this way: 1x, 2x as default, 4 and multiples of 4 till the max. burning speed is reached.

Description

This function reads the max. reading speed of the device given by the selected index.

1.1.1.6.2.39 GetMediumInfo Method

This is the overview for the GetMediumInfo method overload.

Overload List

	Name	Description
	Burner::GetMediumInfo () (see page 42)	This function returns the information about the current medium in the current device according to the MediumInfo (see page 190) structure.
	Burner::GetMediumInfo (int) (see page 42)	This function returns the information about the current medium in the given device (index) according to the MediumInfo (see page 190) structure.

1.1.1.6.2.39.1 Burner::GetMediumInfo Method ()

C++

```
MediumInfo ^ GetMediumInfo();
```

C#

```
public MediumInfo GetMediumInfo();
```

Returns

This function returns the information about the current medium in the current device according to the MediumInfo (see page 190) structure.

Description

This function returns the information about the current medium in the current device according to the MediumInfo (see page 190) structure.

1.1.1.6.2.39.2 Burner::GetMediumInfo Method (int)

C++

```
MediumInfo ^ GetMediumInfo(
    int Index
);
```

C#

```
public MediumInfo GetMediumInfo(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This functions returns a reference to a filled structure MediumInfo (see page 190).

Description

This function returns the information about the current medium in the given device (index) according to the MediumInfo (see page 190) structure.

1.1.1.6.2.40 Burner::GetMpegCount Method

C++

```
int GetMpegCount();
```

C#

```
public int GetMpegCount();
```

Returns

The total number of files in the VCD or SVCD project.

Description

This function calculate the number of added files from a VCD or SVCD project.

1.1.1.6.2.41 Burner::GetPlayTime Method

C++

```
int GetPlayTime(
    String ^ FileName
);
```

C#

```
public int GetPlayTime(
    ref String FileName
);
```

Parameters

Parameters	Description
String ^ FileName	Full file name, with Path, of the audio file.

Returns

The received playtime in seconds.

Description

This function returns the playtime of the given audio file in seconds.

Notes

Use this function only with Ogg, Mp3, WMA, Mp4, Flac and Wav files. Other files will cause an exception.

1.1.1.6.2.42 GetPossibleBurnSpeeds Method

This is the overview for the GetPossibleBurnSpeeds method overload.

Overload List

	Name	Description
	Burner::GetPossibleBurnSpeeds () (see page 44)	This function returns the possible burn speed information of the current medium in the current device.
	Burner::GetPossibleBurnSpeeds (int) (see page 44)	This function returns the possible burn speed information of the current medium of the device according to the given index.

1.1.1.6.2.42.1 Burner::GetPossibleBurnSpeeds Method ()

C++

```
array<Speed ^> ^ GetPossibleBurnSpeeds();
```

C#

```
public array<Speed ^> GetPossibleBurnSpeeds();
```

Returns

This functions returns a reference to a array of speeds according to the Speed (see page 201) structure.

Description

This function returns the possible burn speed information of the current medium in the current device.

Example

```
Speed[] aBurnSpeeds = m_burner.GetPossibleBurnSpeeds();

for (int i = 0; i < aBurnSpeeds.Length; ++i)
{
speedComboBox.Items.Add(
String.Format("{0}x ({1} KB/sec)", aBurnSpeeds[i].SpeedInX,
aBurnSpeeds[i].SpeedInKBPerSec));
}
```

1.1.1.6.2.42.2 Burner::GetPossibleBurnSpeeds Method (int)

C++

```
array<Speed ^> ^ GetPossibleBurnSpeeds(
    int Index
);
```

C#

```
public array<Speed ^> GetPossibleBurnSpeeds(
    int Index
);
```

Returns

This functions returns a reference to a array of speeds according to the Speed (see page 201) structure.

Description

This function returns the possible burn speed information of the current medium of the device according to the given index.

Example

```
Speed[] aBurnSpeeds = m_burner.GetPossibleBurnSpeeds(2);

for (int i = 0; i < aBurnSpeeds.Length; ++i)
{
speedComboBox.Items.Add(
String.Format("{0}x ({1} KB/sec)", aBurnSpeeds[i].SpeedInX,
aBurnSpeeds[i].SpeedInKBPerSec));
}
```

1.1.1.6.2.43 Burner::GetPossibleImageFormats Method

C++

```
ImageFormat GetPossibleImageFormats();
```

C#

```
public ImageFormat GetPossibleImageFormats();
```

Returns

The function will return a value filled according to the ImageFormat (see page 186) enumeration.

Description

This function will check what kind of image formats are possible to save from selected disk.

Example

```
You can read the possible values with a AND query:

ImageFormat cFormats = m_burner.GetPossibleImageFormats();
if ((cFormats & ImageFormat.Iso) == ImageFormat.Iso)
{
// Iso is possible
}
if ((cFormats & ImageFormat.Bin) == ImageFormat.Bin)
{
// Bin is possible
}
```

1.1.1.6.2.44 GetPossibleReadSpeeds Method

This is the overview for the GetPossibleReadSpeeds method overload.

Overload List

	Name	Description
	Burner::GetPossibleReadSpeeds () (see page 45)	This function returns the possible read speed information of the current medium in the current device.
	Burner::GetPossibleReadSpeeds (int) (see page 46)	This function returns the possible read speed information of the current medium of the device according to the given index.

1.1.1.6.2.44.1 Burner::GetPossibleReadSpeeds Method ()

C++

```
array<Speed ^> ^ GetPossibleReadSpeeds();
```

C#

```
public array<Speed ^> GetPossibleReadSpeeds();
```

Returns

This functions returns a reference to a array of speeds according to the Speed (see page 201) structure.

Description

This function returns the possible read speed information of the current medium in the current device.

Example

```
Speed[] aReadSpeeds = m_burner.GetPossibleReadSpeeds();

for (int i = 0; i < aReadSpeeds.Length; ++i)
{
speedComboBox.Items.Add(
String.Format("{0}x ({1} KB/sec)", aReadSpeeds[i].SpeedInX,
aReadSpeeds[i].SpeedInKBPerSec));
}
```

1.1.1.6.2.44.2 Burner::GetPossibleReadSpeeds Method (int)

C++

```
array<Speed ^> ^ GetPossibleReadSpeeds(
    int Index
);
```

C#

```
public array<Speed ^> GetPossibleReadSpeeds(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This functions returns a reference to a array of speeds according to the Speed (see page 201) structure.

Description

This function returns the possible read speed information of the current medium of the device according to the given index.

Example

```
Speed[] aReadSpeeds = m_burner.GetPossibleReadSpeeds(2);

for (int i = 0; i < aReadSpeeds.Length; ++i)
{
speedComboBox.Items.Add(
String.Format("{0}x ({1} KB/sec)", aReadSpeeds[i].SpeedInX,
aReadSpeeds[i].SpeedInKBPerSec));
}
```

1.1.1.6.2.45 Burner::GetPrecisePlayTime Method

C++

```
double GetPrecisePlayTime(
    String ^ FileName
);
```

C#

```
public double GetPrecisePlayTime(
    ref String FileName
);
```

Parameters

Parameters	Description
String ^ FileName	Full file name, with Path, of the audio file.

Returns

The received playtime in milliseconds.

Description

This function returns the playtime of the given audio file in milliseconds.

Notes

Use this function only with Ogg, Mp3, WMA, Mp4, Flac and Wav files. Other files will cause an exception.

1.1.1.6.2.46 Burner::GetProjectType Method

C++

```
ProjectType GetProjectType();
```

C#

```
public ProjectType GetProjectType();
```

Returns

A value filled according to the ProjectType (see page 195) enumeration.

Description

This function returns the type of the current project.

Example

```
You can read the possible values with a AND query:

if (m_burner.GetProjectType() == ProjectType.Data)
{
//Ok, Data Project
}
```

1.1.1.6.2.47 Burner::GetReadSpeed Method

C++

```
int GetReadSpeed(
    int Index
);
```

C#

```
public int GetReadSpeed(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The function will return the read speed.

Description

This Function will get the ReadSpead of the device according to the given index.

1.1.1.6.2.48 GetSessionInfo Method

This is the overview for the GetSessionInfo method overload.

Overload List

	Name	Description
	Burner::GetSessionInfo (int) (see page 48)	This function returns the session information of the current medium in the current device.
	Burner::GetSessionInfo (int, int) (see page 48)	This function returns the session information of the current medium of the device according to the given index.

1.1.1.6.2.48.1 Burner::GetSessionInfo Method (int)

C++

```
SessionInfo ^ GetSessionInfo(
    int Session
);
```

C#

```
public SessionInfo GetSessionInfo(
    int Session
);
```

Parameters

Parameters	Description
int Session	Session index from the session list that you want to check.

Returns

This function returns a reference to a filled SessionInfo (see page 200) structure.

Description

This function returns the session information of the current medium in the current device.

1.1.1.6.2.48.2 Burner::GetSessionInfo Method (int, int)

C++

```
SessionInfo ^ GetSessionInfo(
    int Index,
    int Session
);
```

C#

```
public SessionInfo GetSessionInfo(
    int Index,
    int Session
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int Session	Session index from the session list that you want to check.

Returns

This function returns a reference to a filled SessionInfo (see page 200) structure.

Description

This function returns the session information of the current medium of the device according to the given index.

1.1.1.6.2.49 Burner::GetText Method

C++

```
String ^ GetText(
    int TextID
);
```

C#

```
public String GetText(
    int TextID
);
```

Parameters

Parameters	Description
int TextID	Text-ID of the text you want to get.

Returns

Returns a string that contains the text string. You can set different languages first with SetLanguage (see page 66).

Description

This function returns the text of a TextID, e.g. error code, according to your language setting.

1.1.1.6.2.50 Burner::GetTrackFormatEx Method

C++

```
RawTrackFormat GetTrackFormatEx(
    int nIndex,
    int nTrack
);
```

C#

```
public RawTrackFormat GetTrackFormatEx(
    int nIndex,
    int nTrack
);
```

Parameters

Parameters	Description
int nIndex	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int nTrack	The track index of the current disk to receive information from.

Returns

Returns a type of disk according to a RawTrackFormat (see page 197) enumeration.

Description

This function returns the type of the disk track of the current medium of the given device.

1.1.1.6.2.51 Burner::GetTrackIndexes Method

C++

```
array<int> ^ GetTrackIndexes(
    int Index,
    int Track
);
```

C#

```
public array<int> GetTrackIndexes(
    int Index,
    int Track
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int Track	The index of the track on the disk.

Returns

This function returns a int array filled with the sub indexes.

Description

This function reads a list of sub indexes of the give audio track.

1.1.1.6.2.52 GetTrackInfo Method

This is the overview for the GetTrackInfo method overload.

Overload List

	Name	Description
	Burner::GetTrackInfo (int) (see page 50)	This function returns the track information of the current medium in the current device.
	Burner::GetTrackInfo (int, int) (see page 50)	This function returns the track information of the current medium inside the device according to the give nindex.

1.1.1.6.2.52.1 Burner::GetTrackInfo Method (int)

C++

```
TrackInfo ^ GetTrackInfo(
    int Track
);
```

C#

```
public TrackInfo GetTrackInfo(
    int Track
);
```

Parameters

Parameters	Description
int Track	Track index from the Track list that you want to check

Returns

This functions returns a reference to a filled TrackInfo (see page 203) structure.

Description

This function returns the track information of the current medium in the current device.

1.1.1.6.2.52.2 Burner::GetTrackInfo Method (int, int)

C++

```
TrackInfo ^ GetTrackInfo(
    int Index,
    int Track
);
```

C#

```
public TrackInfo GetTrackInfo(
    int Index,
    int Track
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

int Track	Track index from the Track list that you want to check

Returns

This functions returns a reference to a filled TrackInfo (see page 203) structure.

Description

This function returns the track information of the current medium inside the device according to the give nindex.

1.1.1.6.2.53 Burner::GetTrackISRC Method

C++

```
String ^ GetTrackISRC(
    int Index,
    int Track
);
```

C#

```
public String GetTrackISRC(
    int Index,
    int Track
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int Track	The index of the track on the disk.

Returns

This functions returns a string with the ISRC code. If no ISRC codes was found, the string will be empty.

Description

This function reads the ISRC code from the give audio track.

1.1.1.6.2.54 Burner::GrabAudioTrack Method

C++

```
bool GrabAudioTrack(
    AudioGrabbingParams ^ sParams,
    unsigned int nTrackNumber,
    String ^ strSavePath
);
```

C#

```
public bool GrabAudioTrack(
    ref AudioGrabbingParams sParams,
    unsigned int nTrackNumber,
    ref String strSavePath
);
```

Parameters

Parameters	Description
AudioGrabbingParams ^ sParams	A filled AudioGrabbingParams (see page 143) structure.
unsigned int nTrackNumber	The index of the Track on the disk.
String ^ strSavePath	The full path where to store the file.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

The function will grab/save a Audio Track to a file. Tagging and different Encoding types are available.

1.1.1.6.2.55 Burner::Initialize Method

C++

```
bool Initialize(
    String ^ LicenseKey,
    ASPIInterface AspiInterface,
    bool UseImageDevice
);
```

C#

```
public bool Initialize(
    ref String LicenseKey,
    ASPIInterface AspiInterface,
    bool UseImageDevice
);
```

Parameters

Parameters	Description
String ^ LicenseKey	Pass your license key to be checked. If you provide a valid key, the IsoSDK (see page 1) is activated.
ASPIInterface AspiInterface	With this parameter you can chose the ASPI interface you want to use according to an ASPIInterface (see page 141) enumeration.
bool UseImageDevice	This parameter will enable/disable the ImageWriter inside the Drive list. The ImageWriter is the internal device to create ISO Images from your projects.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function initializes the IsoSDK (see page 1). The function must be called before you can use other functions.

Notes

The function must be called before you can use other functions.

You can only use the internal ASPI layer with Windows 2000 or higher. The internal ASPI layer will not work with Windows98 nor with Windows ME.

1.1.1.6.2.56 IsDeviceReady Method

This is the overview for the IsDeviceReady method overload.

Overload List

	Name	Description
	Burner::IsDeviceReady () (see page 53)	This function tests if the current device is ready.
	Burner::IsDeviceReady (int) (see page 53)	This function tests if the device according to the given index is ready.

1.1.1.6.2.56.1 Burner::IsDeviceReady Method ()

C++

```
bool IsDeviceReady();
```

C#

```
public bool IsDeviceReady();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function tests if the current device is ready.

1.1.1.6.2.56.2 Burner::IsDeviceReady Method (int)

C++

```
bool IsDeviceReady(
    int Index
);
```

C#

```
public bool IsDeviceReady(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function tests if the device according to the given index is ready.

1.1.1.6.2.57 Burner::IsValidVideoTsFolder Method

C++

```
bool IsValidVideoTsFolder(
    String ^ Path
);
```

C#

```
public bool IsValidVideoTsFolder(
    ref String Path
);
```

Parameters

Parameters	Description
String ^ Path	This value contains the full path to the folder where the VideoDVD image is stored. You can include or exclude the VideoTS folder.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (

see page 155)).

Description

This function will check if a given path contains a valid VideoDVD structure with IFO, BUP and VOB Files.

1.1.1.6.2.58 Burner::LoadBassPlugin Method

C++

```
bool LoadBassPlugin(
    String ^ plugFile
);
```

C#

```
public bool LoadBassPlugin(
    ref String plugFile
);
```

Parameters

Parameters	Description
String ^ plugFile	The file to load. File name with full path.

Returns

The function returns true upon successful execution.

Description

This function will load a given Bass plugin to the IsoSDK (see page 1). Bass.dll have to be present.

http://www.un4seen.com/

NOTE: The bass.dll and plugins are available as 32 and 64 bit.

1.1.1.6.2.59 LockMedium Method

This is the overview for the LockMedium method overload.

Overload List

	Name	Description
	Burner::LockMedium (bool) (see page 54)	This function will lock/unlock a medium inside the current device.
	Burner::LockMedium (int, bool) (see page 55)	This function will lock/unlock a medium inside the device according to the given index.

1.1.1.6.2.59.1 Burner::LockMedium Method (bool)

C++

```
bool LockMedium(
    bool Lock
);
```

C#

```
public bool LockMedium(
    bool Lock
);
```

Parameters

Parameters	Description
bool Lock	A bool value to lock (true) or unlock (false) the medium.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will lock/unlock a medium inside the current device.

1.1.1.6.2.59.2 Burner::LockMedium Method (int, bool)

C++

```
bool LockMedium(
    int Index,
    bool Lock
);
```

C#

```
public bool LockMedium(
    int Index,
    bool Lock
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
bool Lock	A bool value to lock (true) or unlock (false) the medium.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will lock/unlock a medium inside the device according to the given index.

1.1.1.6.2.60 OpenDiskSession Method

This is the overview for the OpenDiskSession method overload.

Overload List

	Name	Description
	Burner::OpenDiskSession (int) (see page 55)	This function opens a session on the current medium of the current device.
	Burner::OpenDiskSession (int, FileSystems) (see page 56)	This function opens a session on the current medium of the current device according to the selected File System.
	Burner::OpenDiskSession (int, int, FileSystems) (see page 56)	This function opens a session on the current medium of the device according to the given index.

1.1.1.6.2.60.1 Burner::OpenDiskSession Method (int)

C++

```
DiskSession ^ OpenDiskSession(
    int TrackNumber
);
```

C#

```
public DiskSession OpenDiskSession(
```

```
    int TrackNumber
);
```

Parameters

Parameters	Description
int TrackNumber	Number of the session to be opened.

Returns

This Function will return a reference to a DiskSession (see page 94) class. After you get a reference you can use additional function of the DiskSession (see page 94) class.

Description

This function opens a session on the current medium of the current device.

1.1.1.6.2.60.2 Burner::OpenDiskSession Method (int, FileSystems)

C++

```
DiskSession ^ OpenDiskSession(
    int TrackNumber,
    FileSystems fsType
);
```

C#

```
public DiskSession OpenDiskSession(
    int TrackNumber,
    FileSystems fsType
);
```

Parameters

Parameters	Description
int TrackNumber	Number of the session to be opened.
FileSystems fsType	The file system you want to open according to the FileSystems (see page 186) enumeration.

Returns

This Function will return a reference to a DiskSession (see page 94) class. After you get a reference you can use additional function of the DiskSession (see page 94) class.

Description

This function opens a session on the current medium of the current device according to the selected File System.

1.1.1.6.2.60.3 Burner::OpenDiskSession Method (int, int, FileSystems)

C++

```
DiskSession ^ OpenDiskSession(
    int Index,
    int TrackNumber,
    FileSystems fsType
);
```

C#

```
public DiskSession OpenDiskSession(
    int Index,
    int TrackNumber,
    FileSystems fsType
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int TrackNumber	Number of the session to be opened.
FileSystems fsType	The file system you want to open according to the FileSystems (see page 186) enumeration.

Returns

This Function will return a reference to a DiskSession (see page 94) class. After you get a reference you can use additional function of the DiskSession (see page 94) class.

Description

This function opens a session on the current medium of the device according to the given index.

1.1.1.6.2.61 Burner::PlayAudioFile Method

C++

```
bool PlayAudioFile(
    String ^ AudioFile
);
```

C#

```
public bool PlayAudioFile(
    ref String AudioFile
);
```

Parameters

Parameters	Description
String ^ AudioFile	Full file name, with Path, of the audio file.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

Use this function to play the given file to the default output device.

Notes

Use this function only with Ogg, Mp3, WMA, M4a, AAC and Wav files. Other files will cause an exception.

Make sure that you stop playing a file before you de-init the IsoSDK (see page 1). Otherwise you will get an exception.

1.1.1.6.2.62 Burner::PlayAudioTrack Method

C++

```
bool PlayAudioTrack(
    int nTrackNumber
);
```

C#

```
public bool PlayAudioTrack(
    int nTrackNumber
);
```

Parameters

Parameters	Description
int nTrackNumber	The track number you want to play.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will play a audio track on the current medium or disk image file.

1.1.1.6.2.63 Burner::Prepare Method

C++

```
bool Prepare();
```

C#

```
public bool Prepare();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function prepares the data to be written. In this step the IsoSDK (see page 1) creates the file system and all tables.

Notes

This function maybe need some time to return if you manage many small files on the disc.

1.1.1.6.2.64 Burner::ReadCDText Method

C++

```
CDText ^ ReadCDText(
    int Index
);
```

C#

```
public CDText ReadCDText(
    int Index
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This function returns a CDText (see page 87) handle if success.

Description

This function will create a handle to the CD-Text information of the audio disk according to the selected device.

1.1.1.6.2.65 Burner::ReadSectors Method

C++

```
array<byte> ^ ReadSectors(
    int nIndex,
```

```
    int nLba,
    int nCount,
    ImageFormat Format,
    int buff_length
);
```

C#

```
public array<byte> ReadSectors(
    int nIndex,
    int nLba,
    int nCount,
    ImageFormat Format,
    int buff_length
);
```

Parameters

Parameters	Description
int nIndex	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int nLba	The current LBA to read.
int nCount	Number of tries to read.
ImageFormat Format	Format to read according to the ImageFormat (see page 186) enumeration.
int buff_length	The length of the read buffer. Possible are 2048 for default reading or 2352 for raw reading.

Returns

This Function will return a filled buffer. Length of the buffer is given by buff_length.

Description

This function will read a given sector into a buffer.

1.1.1.6.2.66 Burner::RemoveBurnDevice Method

C++

```
bool RemoveBurnDevice(
    String ^ Device
);
```

C#

```
public bool RemoveBurnDevice(
    ref String Device
);
```

Parameters

Parameters	Description
String ^ Device	The device/drive letter of the device to be removed.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will remove a device from the multi burn device list. This is important for multi device burning.

1.1.1.6.2.67 Burner::RemoveDir Method

C++

```
bool RemoveDir(
```

```
    String ^ DestinationPath
);
```

C#

```
public bool RemoveDir(
    ref String DestinationPath
);
```

Parameters

Parameters	Description
String ^ DestinationPath	A string that names the directory to remove.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function renames/moves a directory in the project.

1.1.1.6.2.68 Burner::RemoveFile Method

C++

```
bool RemoveFile(
    String ^ File,
    String ^ DestinationPath
);
```

C#

```
public bool RemoveFile(
    ref String File,
    ref String DestinationPath
);
```

Parameters

Parameters	Description
String ^ File	File name without path information.
String ^ DestinationPath	Path to the file in the project without file name.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function removes a file from the project.

1.1.1.6.2.69 Burner::RenameDir Method

C++

```
bool RenameDir(
    String ^ SourcePath,
    String ^ DestinationPath
);
```

C#

```
public bool RenameDir(
    ref String SourcePath,
    ref String DestinationPath
);
```

Parameters

Parameters	Description
String ^ SourcePath	A string with the source path of the directory you want to rename/move.
String ^ DestinationPath	A string with the destination path of the directory you want to rename/move.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function renames/moves file in the project.

1.1.1.6.2.70 Burner::RenameFile Method

C++

```
bool RenameFile(
    String ^ SourceFilePath,
    String ^ DestinationFilePath
);
```

C#

```
public bool RenameFile(
    ref String SourceFilePath,
    ref String DestinationFilePath
);
```

Parameters

Parameters	Description
String ^ SourceFilePath	A string with the source path of the file you want to rename/move.
String ^ DestinationFilePath	A string with the destination path of the file you want to rename/move.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function renames/moves directory in the project.

1.1.1.6.2.71 Burner::RescanDevices Method

C++

```
bool RescanDevices();
```

C#

```
public bool RescanDevices();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will rescan the internal device list. This is useful to detect new connected devices like USB and FireWire.

Notes

You can call this function with a timer to check if a new device is connected.

If you use the Windows DeviceChangeEvent you can call this function on this event.

Make sure that you call GetDevices (see page 38)() after this call.

1.1.1.6.2.72 Burner::SaveLogToFile Method

C++

```
bool SaveLogToFile(
    String ^ FilePath
);
```

C#

```
public bool SaveLogToFile(
    ref String FilePath
);
```

Parameters

Parameters	Description
String ^ FilePath	A string that names the file (with full path) where you want to save the log data to.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will save the automatically generated burning log data to a file.

Notes

The IsoSDK (see page 1) will generate a text file if the file does not exist. This data is important for later support requests.

1.1.1.6.2.73 Burner::SaveTrackToFile Method

C++

```
bool SaveTrackToFile(
    int Index,
    int nTrackNumber,
    [In] String ^ FileName,
    SaveTrackFileFormat FileFormat
);
```

C#

```
public bool SaveTrackToFile(
    int Index,
    int nTrackNumber,
    ref [In] String FileName,
    SaveTrackFileFormat FileFormat
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int nTrackNumber	The number of the track inside the disk or image file.
[In] String ^ FileName	The path and name of the target file.

SaveTrackFileFormat FileFormat	The file format of the track according to the SaveTrackFileFormat (see page 200) enumeration.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function saves a selected track to a file.

1.1.1.6.2.74 Burner::SetASPI Method

C++

```
bool SetASPI(
    ASPIInterface AspiInterface
);
```

C#

```
public bool SetASPI(
    ASPIInterface AspiInterface
);
```

Parameters

Parameters	Description
ASPIInterface AspiInterface	An value according to the ASPIInterface (see page 141) enumeration.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

If you want to use the internal ASPI layer after the initialization you use this function to switch from the external to the internal ASPI layer et vice versa .

Notes

After you have called this function you have to set the current project again. All properties and settings of the current project will be deleted.

You can only use the internal ASPI layer with Windows 2000 or higher. The internal ASPI layer will not work with Windows 98 nor with Windows ME.

1.1.1.6.2.75 Burner::SetAudioFileProperty Method

C++

```
bool SetAudioFileProperty(
    AudioFileProperty ^ Property
);
```

C#

```
public bool SetAudioFileProperty(
    ref AudioFileProperty Property
);
```

Parameters

Parameters	Description
AudioFileProperty ^ Property	A filled structure AudioFileProperty (see page 141)

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

The function sets the audio properties for the specified audio file in the project or for the disk. Supported projects: Audio and Mixed Mode.

1.1.1.6.2.76 Burner::SetBurnSpeed Method

C++

```
void SetBurnSpeed(
    int Index,
    int Speed
);
```

C#

```
public SetBurnSpeed(
    int Index,
    int Speed
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int Speed	New Speed (see page 201) to set.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function sets the writing speed of the current device.

Notes

If you use multiple devices (multi-device burning) this option will set the speed for all devices. If a device will not support the speed, it will be set to the nearest speed to this value.

You have to give a valid Speed (see page 201) value to this function. A valid Speed (see page 201) value is a value out of the GetPossibleBurnSpeeds (see page 44) function result.

Please note that there are different values for CD, DVD and Blu-Ray. The IsoSDK (see page 1) can only detect the right values if a blank media is inserted into the target drive.

If no blank media is available the IsoSDK (see page 1) will return the possible Speeds for CD-R.

1.1.1.6.2.77 Burner::SetFileAttr Method

C++

```
bool SetFileAttr(
    String ^ filePath,
    FileAttributes fileAttributes
);
```

C#

```
public bool SetFileAttr(
```

```
    ref String filePath,
    FileAttributes fileAttributes
);
```

Parameters

Parameters	Description
String ^ filePath	File name + full path of the file to the the attribute to.
FileAttributes fileAttributes	New attribute according to the enumeration FileAttributes (see page 184)

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function set the attributes to a file.

1.1.1.6.2.78 Burner::SetFileTimes Method

C++

```
bool SetFileTimes(
    String ^ filePath,
    FileDateTime ^ creationTime,
    FileDateTime ^ modificationTime,
    FileDateTime ^ accessTime
);
```

C#

```
public bool SetFileTimes(
    ref String filePath,
    ref FileDateTime creationTime,
    ref FileDateTime modificationTime,
    ref FileDateTime accessTime
);
```

Parameters

Parameters	Description
String ^ filePath	File name + full path of the file to the the value to.
FileDateTime ^ creationTime	Creation time of the file.
FileDateTime ^ modificationTime	Modification time of the file.
FileDateTime ^ accessTime	Last access time of the file.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will set the custom dates to a file.

1.1.1.6.2.79 Burner::SetFileUserParam Method

C++

```
bool SetFileUserParam(
    String ^ filePath,
    void * userParam
);
```

C#

```
public bool SetFileUserParam(
    ref String filePath,
```

```
    ref  userParam
);
```

Parameters

Parameters	Description
String ^ filePath	File name + full path of the file to the the value to.
void * userParam	A pointer to a value or custom structure.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This is a data value you can set yourself according to your needs. You can pass a integer or structure to the file that stores extra data.

Notes

The userParam was added for sorting feature but you can also use it for other function in your own code.

1.1.1.6.2.80 Burner::SetImageFilePath Method

C++

```
bool SetImageFilePath(
    String ^ Path
);
```

C#

```
public bool SetImageFilePath(
    ref String Path
);
```

Parameters

Parameters	Description
String ^ Path	File name + full path of the image file.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

The function sets the path to the image (*.iso) file you want to burn using the 'Imagewriter'.

1.1.1.6.2.81 Burner::SetLanguage Method

C++

```
bool SetLanguage(
    String ^ Language,
    String ^ LanguageFile
);
```

C#

```
public bool SetLanguage(
    ref String Language,
    ref String LanguageFile
);
```

Parameters

Parameters	Description
String ^ Language	Language (see page 74). This is the language section in the INI file (e.g. [English]).
String ^ LanguageFile	File name of the language ini file (if you do not provide a folder, the SDK will look for the specified file in the program folder).

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function sets the language of the IsoSDK (see page 1). The following resources are effected: error codes, message codes and GUI codes.

Notes

Set the language of the IsoSDK (see page 1) as early as possible.

1.1.1.6.2.82 Burner::SetRawStructure Method

C++

```
bool SetRawStructure(
    array<RawTrack ^> ^ TrackList
);
```

C#

```
public bool SetRawStructure(
    ref array<RawTrack ^> TrackList
);
```

Parameters

Parameters	Description
array<RawTrack ^> ^ TrackList	An array of RawTrack (see page 196) structures.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

The function sets the structure of the disc and the format of the input data for a RAW project.

1.1.1.6.2.83 Burner::SetReadSpeed Method

C++

```
void SetReadSpeed(
    int Index,
    int Speed
);
```

C#

```
public SetReadSpeed(
    int Index,
    int Speed
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
int Speed	The new read speed value. If you only want to use the max. reading speed, pass the value BS_MAX_SPEED. This is the default value.

Returns

No return value available.

Description

This function sets the reading speed of the device according to the given index.

Notes

You have to give a valid Speed (see page 201) value to this function. A valid Speed (see page 201) value is a value out of the GetPossibleReadSpeeds (see page 45) function result.

Please note that there are different values for CD, DVD and Blu-Ray. The IsoSDK (see page 1) can only detect the right values if a blank media is inserted into the target drive.

If no blank media is available the IsoSDK (see page 1) will return the possible Speeds for CD-R.

1.1.1.6.2.84 SetRegionalCode Method

This is the overview for the SetRegionalCode method overload.

Overload List

	Name	Description
	Burner::SetRegionalCode (int) (see page 68)	This function will change the region code of a device. You can set 1-6. 7 and 8 are not allowed to set. You also can not set back to "unset" mode 0.
	Burner::SetRegionalCode (int, [In] int) (see page 69)	This function will change the region code of the device according to the given index. You can set 1-6. 7 and 8 are not allowed to set. You also can not set back to "unset" mode 0.

1.1.1.6.2.84.1 Burner::SetRegionalCode Method (int)

C++

```
bool SetRegionalCode(
    int RPC
);
```

C#

```
public bool SetRegionalCode(
    int RPC
);
```

Parameters

Parameters	Description
int RPC	The new region code to be set.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will change the region code of a device. You can set 1-6. 7 and 8 are not allowed to set. You also can not set back to "unset" mode 0.

Notes

The change of a region code is limited! So if you allow this function in your software make sure to inform the user about this.

1.1.1.6.2.84.2 Burner::SetRegionalCode Method (int, [In] int)

C++

```
bool SetRegionalCode(
    int Index,
    [In] int RPC
);
```

C#

```
public bool SetRegionalCode(
    int Index,
    [In] int RPC
);
```

Parameters

Parameters	Description
int Index	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.
[In] int RPC	The new region code to be set.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function will change the region code of the device according to the given index. You can set 1-6. 7 and 8 are not allowed to set. You also can not set back to "unset" mode 0.

Notes

The change of a region code is limited! So if you allow this function in your software make sure to inform the user about this.

1.1.1.6.2.85 Burner::SetVCDKeyHandler Method

C++

```
int SetVCDKeyHandler(
    int nItemNumber,
    int nKey,
    int nNextItemNumber
);
```

C#

```
public int SetVCDKeyHandler(
    int nItemNumber,
    int nKey,
    int nNextItemNumber
);
```

Parameters

Parameters	Description
int nItemNumber	Item number to set the key handler to.
int nKey	ID of the pressed key. Following IDs are allowed:

int nNextItemNumber	Item number of the next item.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function sets which item will be shown next when particular key is pressed.

1.1.1.6.2.86 Burner::SetVCDTimeOutHandler Method

C++

```
int SetVCDTimeOutHandler(
    int nItemNumber,
    int nTimeOut,
    int nNextItemNumber
);
```

C#

```
public int SetVCDTimeOutHandler(
    int nItemNumber,
    int nTimeOut,
    int nNextItemNumber
);
```

Parameters

Parameters	Description
int nItemNumber	Item number to set the timeout to.
int nTimeOut	Timeout value.
int nNextItemNumber	Item number of the next item.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function sets transition timeout between items.

1.1.1.6.2.87 Burner::StopMpegAction Method

C++

```
bool StopMpegAction();
```

C#

```
public bool StopMpegAction();
```

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function kills the current scanning process of an added video file to a VCD or SVCD project.

Notes

This function immediately stops the scan. No data is available for burning.

1.1.1.6.2.88 Burner::TagsFromNetworkDialog Method

C++

```
NetworkTags ^ TagsFromNetworkDialog(
    int nIndex
);
```

C#

```
public NetworkTags TagsFromNetworkDialog(
    int nIndex
);
```

Parameters

Parameters	Description
int nIndex	The index of the device you want to get the information from. Call GetActiveDevicesCount (see page 33)() to get the device count.

Returns

This function returns a NetworkTag handle if success.

Description

This function will create a handle to the NetworkTags (see page 122) information of the audio disk according to the selected device.

1.1.1.6.3 Burner Properties

The properties of the Burner class are listed here.

Public Properties

	Name	Description
R	BootOptions (see page 72)	This property is a pointer to the IsoSDK::BootOptions (see page 10) Class to set/get BootDisk properties / settings.
	BurnDevice (see page 72)	This property sets and gets the current burn device. This drive will be used for the burn process.
	BurnSpeed (see page 72)	This property set and get the writing speed of the current device.
R	CompressEncryptOptions (see page 73)	This property is a pointer to the IsoSDK::CompressEncryptOptions (see page 90) Class to set/get properties / settings for encryption and compressing.
R	DVDVideoOptions (see page 73)	This property is a pointer to the IsoSDK::DVDVideoOptions (see page 98) Class to set/get extended VideoDVD properties / settings.
R	FileDateTimeEx (see page 73)	This is FileDateTimeEx, a member of class Burner.
R	FirstSegmentIndex (see page 73)	Represents the first segment to play on a VCDKeyHandler. Default starts at 1000.
R	FirstTrackIndex (see page 73)	Represents the first track to play on a VCDKeyHandler. Default starts at 0.
	ImageDeviceEnabled (see page 74)	This property set and get the state of the internal ImageDevice switch.
R	ISOExOptions (see page 74)	This property is a pointer to the IsoSDK::ISOExOptions (see page 114) Class to set/get ISO Ex options properties / settings.
	Language (see page 74)	This property set and get the language of the IsoSDK (see page 1). The following resources are effected: error codes, message codes and GUI codes.
R	Options (see page 74)	This property is a pointer to the IsoSDK::GeneralOptions (see page 107) Class to set/get general options properties / settings.
	ReadDevice (see page 74)	This property sets and gets the current read device. This drive will be used for the read process.

	ReadSpeed (see page 75)	This property set and get the reading speed of the current device.
	TmpPath (see page 75)	This property sets and gets the reading speed of the current device.
R	UDFOptions (see page 75)	This property is a pointer to the IsoSDK::UDFOptions (see page 128) Class to set/get UDFOptions properties / settings.
R	VCDInfiniteTimeout (see page 75)	This property represents the timeout value used in SetVCDTimeOutHandler (see page 70) method.
R	VCDKey0 (see page 75)	User interaction (Button). disabled with $FFFD. Else it will play the selected item in the menu selection.
R	VCDKeyDefault (see page 76)	User interaction (Button). This item stops current playback or wait state and returns to the default list item or given item in play list.
R	VCDKeyNext (see page 76)	User interaction (Button). This item stops current playback or wait state and plays the next item in the play list.
R	VCDKeyPrevious (see page 76)	User interaction (Button). This item stops current playback or wait state and plays the previous item in the play list.
R	VCDKeyReturn (see page 76)	User interaction (Button). This item stops current playback or wait state and returns to menu or given item in play list.
	Verify (see page 76)	Use this property to get and set the after burn verification state.
	WriteCDTextInUnicode (see page 77)	Get/Set the property to write CD-Text with Unicode or Multibyte.

1.1.1.6.3.1 Burner::BootOptions Property

C++

```
property BootOptions ^ BootOptions;
```

C#

```
public BootOptions BootOptions;
```

Description

This property is a pointer to the IsoSDK::BootOptions (see page 10) Class to set/get BootDisk properties / settings.

Notes

All settings are part of the “El Torito” Specification. Please look inside it to learn how to use the settings.

These settings will be reset after a new call of a data project.

1.1.1.6.3.2 Burner::BurnDevice Property

C++

```
property String ^ BurnDevice;
```

C#

```
public String BurnDevice;
```

Description

This property sets and gets the current burn device. This drive will be used for the burn process.

1.1.1.6.3.3 Burner::BurnSpeed Property

C++

```
property int BurnSpeed;
```

C#

```
public int BurnSpeed;
```

Description

This property set and get the writing speed of the current device.

1.1.1.6.3.4 Burner::CompressEncryptOptions Property

C++

```
property CompressEncryptOptions ^ CompressEncryptOptions;
```

C#

```
public CompressEncryptOptions CompressEncryptOptions;
```

Description

This property is a pointer to the IsoSDK::CompressEncryptOptions (see page 90) Class to set/get properties / settings for encryption and compressing.

1.1.1.6.3.5 Burner::DVDVideoOptions Property

C++

```
property DVDVideoOptions ^ DVDVideoOptions;
```

C#

```
public DVDVideoOptions DVDVideoOptions;
```

Description

This property is a pointer to the IsoSDK::DVDVideoOptions (see page 98) Class to set/get extended VideoDVD properties / settings.

1.1.1.6.3.6 Burner::FileDateTimeEx Property

C++

```
property FileDateTimeEx^ FileDateTimeEx;
```

C#

```
public FileDateTimeEx FileDateTimeEx;
```

Description

This is FileDateTimeEx, a member of class Burner.

1.1.1.6.3.7 Burner::FirstSegmentIndex Property

C++

```
property int FirstSegmentIndex;
```

C#

```
public int FirstSegmentIndex;
```

Description

Represents the first segment to play on a VCDKeyHandler. Default starts at 1000.

1.1.1.6.3.8 Burner::FirstTrackIndex Property

C++

```
property int FirstTrackIndex;
```

C#

```
public int FirstTrackIndex;
```

Description

Represents the first track to play on a VCDKeyHandler. Default starts at 0.

1.1.1.6.3.9 Burner::ImageDeviceEnabled Property

C++

```
property bool ImageDeviceEnabled;
```

C#

```
public bool ImageDeviceEnabled;
```

Description

This property set and get the state of the internal ImageDevice switch.

1.1.1.6.3.10 Burner::ISOExOptions Property

C++

```
property ISOExOptions ^ ISOExOptions;
```

C#

```
public ISOExOptions ISOExOptions;
```

Description

This property is a pointer to the IsoSDK::ISOExOptions (see page 114) Class to set/get ISO Ex options properties / settings.

1.1.1.6.3.11 Burner::Language Property

C++

```
property String ^ Language;
```

C#

```
public String Language;
```

Description

This property set and get the language of the IsoSDK (see page 1). The following resources are effected: error codes, message codes and GUI codes.

1.1.1.6.3.12 Burner::Options Property

C++

```
property GeneralOptions ^ Options;
```

C#

```
public GeneralOptions Options;
```

Description

This property is a pointer to the IsoSDK::GeneralOptions (see page 107) Class to set/get general options properties / settings.

1.1.1.6.3.13 Burner::ReadDevice Property

C++

```
property String ^ ReadDevice;
```

C#

```
public String ReadDevice;
```

Description

This property sets and gets the current read device. This drive will be used for the read process.

1.1.1.6.3.14 Burner::ReadSpeed Property

C++

```
property int ReadSpeed;
```

C#

```
public int ReadSpeed;
```

Description

This property set and get the reading speed of the current device.

1.1.1.6.3.15 Burner::TmpPath Property

C++

```
property String ^ TmpPath;
```

C#

```
public String TmpPath;
```

Description

This property sets and gets the reading speed of the current device.

1.1.1.6.3.16 Burner::UDFOptions Property

C++

```
property UDFOptions ^ UDFOptions;
```

C#

```
public UDFOptions UDFOptions;
```

Description

This property is a pointer to the IsoSDK::UDFOptions (see page 128) Class to set/get UDFOptions properties / settings.

1.1.1.6.3.17 Burner::VCDInfiniteTimeout Property

C++

```
property int VCDInfiniteTimeout;
```

C#

```
public int VCDInfiniteTimeout;
```

Description

This property represents the timeout value used in SetVCDTimeOutHandler (see page 70) method.

1.1.1.6.3.18 Burner::VCDKey0 Property

C++

```
property int VCDKey0;
```

C#

```
public int VCDKey0;
```

Description

User interaction (Button). disabled with $FFFD. Else it will play the selected item in the menu selection.

1.1.1.6.3.19 Burner::VCDKeyDefault Property

C++

```
property int VCDKeyDefault;
```

C#

```
public int VCDKeyDefault;
```

Description

User interaction (Button). This item stops current playback or wait state and returns to the default list item or given item in play list.

1.1.1.6.3.20 Burner::VCDKeyNext Property

C++

```
property int VCDKeyNext;
```

C#

```
public int VCDKeyNext;
```

Description

User interaction (Button). This item stops current playback or wait state and plays the next item in the play list.

1.1.1.6.3.21 Burner::VCDKeyPrevious Property

C++

```
property int VCDKeyPrevious;
```

C#

```
public int VCDKeyPrevious;
```

Description

User interaction (Button). This item stops current playback or wait state and plays the previous item in the play list.

1.1.1.6.3.22 Burner::VCDKeyReturn Property

C++

```
property int VCDKeyReturn;
```

C#

```
public int VCDKeyReturn;
```

Description

User interaction (Button). This item stops current playback or wait state and returns to menu or given item in play list.

1.1.1.6.3.23 Burner::Verify Property

C++

```
property bool Verify;
```

C#

```
public bool Verify;
```

Description

Use this property to get and set the after burn verification state.

1.1.1.6.3.24 Burner::WriteCDTextInUnicode Property

C++

```
property bool WriteCDTextInUnicode;
```

C#

```
public bool WriteCDTextInUnicode;
```

Description

Get/Set the property to write CD-Text with Unicode or Multibyte.

1.1.1.6.4 Burner Delegates

The delegates of the Burner class are listed here.

Public Delegates

Name	Description
AddFileEventHandler (see page 77)	This is nested type IsoSDK::Burner::AddFileEventHandler.
AudioDecodeDoneEventHandler (see page 78)	This is nested type IsoSDK::Burner::AudioDecodeDoneEventHandler.
AudioDecoderEventHandler (see page 78)	This is nested type IsoSDK::Burner::AudioDecoderEventHandler.
BurnDoneEventHandler (see page 78)	This is nested type IsoSDK::Burner::BurnDoneEventHandler.
BurnFileEventHandler (see page 78)	This is nested type IsoSDK::Burner::BurnFileEventHandler.
CompareFilesForArrangementHandler (see page 78)	This is nested type IsoSDK::Burner::CompareFilesForArrangementHandler.
CreateDirEventHandler (see page 79)	This is nested type IsoSDK::Burner::CreateDirEventHandler.
EraseDoneEventHandler (see page 79)	This is nested type IsoSDK::Burner::EraseDoneEventHandler.
FinalizeEventHandler (see page 79)	This is nested type IsoSDK::Burner::FinalizeEventHandler.
InfoTextEventHandler (see page 79)	This is nested type IsoSDK::Burner::InfoTextEventHandler.
JobDoneEventHandler (see page 79)	This is nested type IsoSDK::Burner::JobDoneEventHandler.
ProcessEventHandler (see page 80)	This is nested type IsoSDK::Burner::ProcessEventHandler.
RemoveFileEventHandler (see page 80)	This is nested type IsoSDK::Burner::RemoveFileEventHandler.
StartVerifyEventHandler (see page 80)	This is nested type IsoSDK::Burner::StartVerifyEventHandler.
TextEventHandler (see page 80)	This is nested type IsoSDK::Burner::TextEventHandler.
VerifyDoneEventHandler (see page 80)	This is nested type IsoSDK::Burner::VerifyDoneEventHandler.
VerifyErrorEventHandler (see page 81)	This is nested type IsoSDK::Burner::VerifyErrorEventHandler.
VerifyFileEventHandler (see page 81)	This is nested type IsoSDK::Burner::VerifyFileEventHandler.
VerifySectorEventHandler (see page 81)	This is nested type IsoSDK::Burner::VerifySectorEventHandler.
VideoScanDoneEventHandler (see page 81)	This is nested type IsoSDK::Burner::VideoScanDoneEventHandler.
VideoScannerEventHandler (see page 81)	This is nested type IsoSDK::Burner::VideoScannerEventHandler.

1.1.1.6.4.1 Burner::AddFileEventHandler Delegate

C++

```
delegate void AddFileEventHandler(Object ^sender, AddFileEventArgs ^e);
```

C#

```
public delegate AddFileEventHandler();
```

Description

This is nested type IsoSDK::Burner::AddFileEventHandler.

1.1.1.6.4.2 Burner::AudioDecodeDoneEventHandler Delegate

C++

```
delegate void AudioDecodeDoneEventHandler(Object ^sender, AudioDecodeDoneEventArgs ^e);
```

C#

```
public delegate AudioDecodeDoneEventHandler();
```

Description

This is nested type IsoSDK::Burner::AudioDecodeDoneEventHandler.

1.1.1.6.4.3 Burner::AudioDecoderEventHandler Delegate

C++

```
delegate void AudioDecoderEventHandler(Object ^sender, AudioDecoderEventArgs ^e);
```

C#

```
public delegate AudioDecoderEventHandler();
```

Description

This is nested type IsoSDK::Burner::AudioDecoderEventHandler.

1.1.1.6.4.4 Burner::BurnDoneEventHandler Delegate

C++

```
delegate void BurnDoneEventHandler(Object ^sender, BurnDoneEventArgs ^e);
```

C#

```
public delegate BurnDoneEventHandler();
```

Description

This is nested type IsoSDK::Burner::BurnDoneEventHandler.

1.1.1.6.4.5 Burner::BurnFileEventHandler Delegate

C++

```
delegate void BurnFileEventHandler(Object ^sender, BurnFileEventArgs ^e);
```

C#

```
public delegate BurnFileEventHandler();
```

Description

This is nested type IsoSDK::Burner::BurnFileEventHandler.

1.1.1.6.4.6 Burner::CompareFilesForArrangementHandler Delegate

C++

```
delegate bool CompareFilesForArrangementHandler(Object ^sender,
CompareFilesForArrangementEventArgs ^e);
```

C#

```
public delegate CompareFilesForArrangementHandler();
```

Description

This is nested type IsoSDK::Burner::CompareFilesForArrangementHandler.

1.1.1.6.4.7 Burner::CreateDirEventHandler Delegate

C++

```
delegate void CreateDirEventHandler(Object ^sender, CreateDirEventArgs ^e);
```

C#

```
public delegate CreateDirEventHandler();
```

Description

This is nested type IsoSDK::Burner::CreateDirEventHandler.

1.1.1.6.4.8 Burner::EraseDoneEventHandler Delegate

C++

```
delegate void EraseDoneEventHandler(Object ^sender, EraseDoneEventArgs ^e);
```

C#

```
public delegate EraseDoneEventHandler();
```

Description

This is nested type IsoSDK::Burner::EraseDoneEventHandler.

1.1.1.6.4.9 Burner::FinalizeEventHandler Delegate

C++

```
delegate void FinalizeEventHandler(Object ^sender, EventArgs ^e);
```

C#

```
public delegate FinalizeEventHandler();
```

Description

This is nested type IsoSDK::Burner::FinalizeEventHandler.

1.1.1.6.4.10 Burner::InfoTextEventHandler Delegate

C++

```
delegate void InfoTextEventHandler(Object ^sender, InfoTextEventArgs ^e);
```

C#

```
public delegate InfoTextEventHandler();
```

Description

This is nested type IsoSDK::Burner::InfoTextEventHandler.

1.1.1.6.4.11 Burner::JobDoneEventHandler Delegate

C++

```
delegate void JobDoneEventHandler(Object ^sender, EventArgs ^e);
```

C#

```
public delegate JobDoneEventHandler();
```

Description

This is nested type IsoSDK::Burner::JobDoneEventHandler.

1.1.1.6.4.12 Burner::ProcessEventHandler Delegate

C++

```
delegate void ProcessEventHandler(Object ^sender, ProcessEventArgs ^e);
```

C#

```
public delegate ProcessEventHandler();
```

Description

This is nested type IsoSDK::Burner::ProcessEventHandler.

1.1.1.6.4.13 Burner::RemoveFileEventHandler Delegate

C++

```
delegate void RemoveFileEventHandler(Object ^sender, RemoveFileEventArgs ^e);
```

C#

```
public delegate RemoveFileEventHandler();
```

Description

This is nested type IsoSDK::Burner::RemoveFileEventHandler.

1.1.1.6.4.14 Burner::StartVerifyEventHandler Delegate

C++

```
delegate void StartVerifyEventHandler(Object ^sender, EventArgs ^e);
```

C#

```
public delegate StartVerifyEventHandler();
```

Description

This is nested type IsoSDK::Burner::StartVerifyEventHandler.

1.1.1.6.4.15 Burner::TextEventHandler Delegate

C++

```
delegate void TextEventHandler(Object ^sender, TextEventArgs ^e);
```

C#

```
public delegate TextEventHandler();
```

Description

This is nested type IsoSDK::Burner::TextEventHandler.

1.1.1.6.4.16 Burner::VerifyDoneEventHandler Delegate

C++

```
delegate void VerifyDoneEventHandler(Object ^sender, VerifyDoneEventArgs ^e);
```

C#

```
public delegate VerifyDoneEventHandler();
```

Description

This is nested type IsoSDK::Burner::VerifyDoneEventHandler.

1.1.1.6.4.17 Burner::VerifyErrorEventHandler Delegate

C++

```
delegate void VerifyErrorEventHandler(Object ^sender, VerifyErrorEventArgs ^e);
```

C#

```
public delegate VerifyErrorEventHandler();
```

Description

This is nested type IsoSDK::Burner::VerifyErrorEventHandler.

1.1.1.6.4.18 Burner::VerifyFileEventHandler Delegate

C++

```
delegate void VerifyFileEventHandler(Object ^sender, VerifyFileEventArgs ^e);
```

C#

```
public delegate VerifyFileEventHandler();
```

Description

This is nested type IsoSDK::Burner::VerifyFileEventHandler.

1.1.1.6.4.19 Burner::VerifySectorEventHandler Delegate

C++

```
delegate void VerifySectorEventHandler(Object ^sender, VerifySectorEventArgs ^e);
```

C#

```
public delegate VerifySectorEventHandler();
```

Description

This is nested type IsoSDK::Burner::VerifySectorEventHandler.

1.1.1.6.4.20 Burner::VideoScanDoneEventHandler Delegate

C++

```
delegate void VideoScanDoneEventHandler(Object ^sender, VideoScanDoneEventArgs ^e);
```

C#

```
public delegate VideoScanDoneEventHandler();
```

Description

This is nested type IsoSDK::Burner::VideoScanDoneEventHandler.

1.1.1.6.4.21 Burner::VideoScannerEventHandler Delegate

C++

```
delegate void VideoScannerEventHandler(Object ^sender, VideoScannerEventArgs ^e);
```

C#

```
public delegate VideoScannerEventHandler();
```

Description

This is nested type IsoSDK::Burner::VideoScannerEventHandler.

1.1.1.6.5 Burner Events

The events of the Burner class are listed here.

Public Events

	Name	Description
	AddFileEvent (see page 82)	This event will be triggered when a file was successfully added to the project.
	AudioDecodeDoneEvent (see page 83)	This event occurs when the process of decoding was completed.
	AudioDecoderEvent (see page 83)	This event will be triggered while the process of decoding.
	BurnDoneEvent (see page 83)	This event will be triggered when the burning process was completed.
	BurnFileEvent (see page 83)	This event will be triggered for each file in the current burning process.
	CompareFilesForArrangementEvent (see page 83)	This event will will compare file 1 with file 2 and give the status if file1 will be added before file 2. This callback is needed for sorting a disk.
	CreateDirEvent (see page 84)	This event will be triggered after a new directory was created in the current project.
	EraseDoneEvent (see page 84)	This event will be triggered when the deletion process was completed.
	FinalizeEvent (see page 84)	This event will be triggered during the burning process when the finalize process starts. After this event is triggered the Event ProcessEvent (see page 85) will report the progress of the finalize process.
	InfoTextEvent (see page 84)	This event is triggered when the IsoSDK (see page 1) was loaded and an action was performed. This event is useable for logging.
	JobDoneEvent (see page 84)	This event will be triggered when all operations with the device are finished (after burn or erase process) and the IsoSDK (see page 1) is available for further use.
	ProcessEvent (see page 85)	This event will be triggered during a deletion or burn, erase and finalize process. It shows the current progress of the process.
	RemoveFileEvent (see page 85)	This event will be triggered when a file was removed from the project.
	StartVerifyEvent (see page 85)	This event will be triggered when the verify process will start.
	TextEvent (see page 85)	This event will be triggered when you call the GetText (see page 48)() function.
	VerifyDoneEvent (see page 85)	This event will be triggered when the verify process is finished.
	VerifyErrorEvent (see page 86)	This event is triggered if an error occurred during the current verify process.
	VerifyFileEvent (see page 86)	This event will be triggered every time a new file will be verified.
	VerifySectorEvent (see page 86)	This event will be triggered while sector verify (CreateImage (see page 29) / CopyDisc).
	VideoScanDoneEvent (see page 86)	This event will be triggered when the video scanning process is completed.
	VideoScannerEvent (see page 86)	This event will be triggered during a running scanning process. It shows the current progress of the process.

1.1.1.6.5.1 Burner::AddFileEvent Event

C++

```
event AddFileEventHandler ^ AddFileEvent;
```

C#

```
public event AddFileEventHandler AddFileEvent;
```

Description

This event will be triggered when a file was successfully added to the project.

1.1.1.6.5.2 Burner::AudioDecodeDoneEvent Event

C++

```
event AudioDecodeDoneEventHandler ^ AudioDecodeDoneEvent;
```

C#

```
public event AudioDecodeDoneEventHandler AudioDecodeDoneEvent;
```

Description

This event occurs when the process of decoding was completed.

1.1.1.6.5.3 Burner::AudioDecoderEvent Event

C++

```
event AudioDecoderEventHandler ^ AudioDecoderEvent;
```

C#

```
public event AudioDecoderEventHandler AudioDecoderEvent;
```

Description

This event will be triggered while the process of decoding.

1.1.1.6.5.4 Burner::BurnDoneEvent Event

C++

```
event BurnDoneEventHandler ^ BurnDoneEvent;
```

C#

```
public event BurnDoneEventHandler BurnDoneEvent;
```

Description

This event will be triggered when the burning process was completed.

1.1.1.6.5.5 Burner::BurnFileEvent Event

C++

```
event BurnFileEventHandler ^ BurnFileEvent;
```

C#

```
public event BurnFileEventHandler BurnFileEvent;
```

Description

This event will be triggered for each file in the current burning process.

1.1.1.6.5.6 Burner::CompareFilesForArrangementEvent Event

C++

```
event CompareFilesForArrangementHandler ^ CompareFilesForArrangementEvent;
```

C#

```
public event CompareFilesForArrangementHandler CompareFilesForArrangementEvent;
```

Description

This event will will compare file 1 with file 2 and give the status if file1 will be added before file 2. This callback is needed for sorting a disk.

1.1.1.6.5.7 Burner::CreateDirEvent Event

C++

```
event CreateDirEventHandler ^ CreateDirEvent;
```

C#

```
public event CreateDirEventHandler CreateDirEvent;
```

Description

This event will be triggered after a new directory was created in the current project.

1.1.1.6.5.8 Burner::EraseDoneEvent Event

C++

```
event EraseDoneEventHandler ^ EraseDoneEvent;
```

C#

```
public event EraseDoneEventHandler EraseDoneEvent;
```

Description

This event will be triggered when the deletion process was completed.

1.1.1.6.5.9 Burner::FinalizeEvent Event

C++

```
event FinalizeEventHandler ^ FinalizeEvent;
```

C#

```
public event FinalizeEventHandler FinalizeEvent;
```

Description

This event will be triggered during the burning process when the finalize process starts. After this event is triggered the Event ProcessEvent (see page 85) will report the progress of the finalize process.

1.1.1.6.5.10 Burner::InfoTextEvent Event

C++

```
event InfoTextEventHandler ^ InfoTextEvent;
```

C#

```
public event InfoTextEventHandler InfoTextEvent;
```

Description

This event is triggered when the IsoSDK (see page 1) was loaded and an action was performed. This event is useable for logging.

1.1.1.6.5.11 Burner::JobDoneEvent Event

C++

```
event JobDoneEventHandler ^ JobDoneEvent;
```

C#

```
public event JobDoneEventHandler JobDoneEvent;
```

Description

This event will be triggered when all operations with the device are finished (after burn or erase process) and the IsoSDK (see page 1) is available for further use.

1.1.1.6.5.12 Burner::ProcessEvent Event

C++

```
event ProcessEventHandler ^ ProcessEvent;
```

C#

```
public event ProcessEventHandler ProcessEvent;
```

Description

This event will be triggered during a deletion or burn, erase and finalize process. It shows the current progress of the process.

1.1.1.6.5.13 Burner::RemoveFileEvent Event

C++

```
event RemoveFileEventHandler ^ RemoveFileEvent;
```

C#

```
public event RemoveFileEventHandler RemoveFileEvent;
```

Description

This event will be triggered when a file was removed from the project.

1.1.1.6.5.14 Burner::StartVerifyEvent Event

C++

```
event StartVerifyEventHandler ^ StartVerifyEvent;
```

C#

```
public event StartVerifyEventHandler StartVerifyEvent;
```

Description

This event will be triggered when the verify process will start.

1.1.1.6.5.15 Burner::TextEvent Event

C++

```
event TextEventHandler ^ TextEvent;
```

C#

```
public event TextEventHandler TextEvent;
```

Description

This event will be triggered when you call the GetText (see page 48)() function.

1.1.1.6.5.16 Burner::VerifyDoneEvent Event

C++

```
event VerifyDoneEventHandler ^ VerifyDoneEvent;
```

C#

```
public event VerifyDoneEventHandler VerifyDoneEvent;
```

Description

This event will be triggered when the verify process is finished.

1.1.1.6.5.17 Burner::VerifyErrorEvent Event

C++

```
event VerifyErrorEventHandler ^ VerifyErrorEvent;
```

C#

```
public event VerifyErrorEventHandler VerifyErrorEvent;
```

Description

This event is triggered if an error occurred during the current verify process.

1.1.1.6.5.18 Burner::VerifyFileEvent Event

C++

```
event VerifyFileEventHandler ^ VerifyFileEvent;
```

C#

```
public event VerifyFileEventHandler VerifyFileEvent;
```

Description

This event will be triggered every time a new file will be verified.

1.1.1.6.5.19 Burner::VerifySectorEvent Event

C++

```
event VerifySectorEventHandler ^ VerifySectorEvent;
```

C#

```
public event VerifySectorEventHandler VerifySectorEvent;
```

Description

This event will be triggered while sector verify (CreateImage (see page 29) / CopyDisc).

1.1.1.6.5.20 Burner::VideoScanDoneEvent Event

C++

```
event VideoScanDoneEventHandler ^ VideoScanDoneEvent;
```

C#

```
public event VideoScanDoneEventHandler VideoScanDoneEvent;
```

Description

This event will be triggered when the video scanning process is completed.

1.1.1.6.5.21 Burner::VideoScannerEvent Event

C++

```
event VideoScannerEventHandler ^ VideoScannerEvent;
```

C#

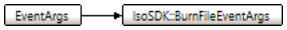
```
public event VideoScannerEventHandler VideoScannerEvent;
```

Description

This event will be triggered during a running scanning process. It shows the current progress of the process.

1.1.1.7 BurnFileEventArgs Class

Class Hierarchy



C++

```
ref class BurnFileEventArgs : public EventArgs;
```

C#

```
public class BurnFileEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the BurnFile event.

1.1.1.7.1 BurnFileEventArgs Members

The following tables list the members exposed by BurnFileEventArgs.

Public Data Members

	Name	Description
	m_strFileName (see page 87)	A string with the source path of the burned file (includes file name).

1.1.1.7.2 BurnFileEventArgs Data Members

The data members of the BurnFileEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strFileName (see page 87)	A string with the source path of the burned file (includes file name).

1.1.1.7.2.1 BurnFileEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

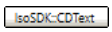
```
public String m_strFileName;
```

Description

A string with the source path of the burned file (includes file name).

1.1.1.8 CDText Class

Class Hierarchy



C++

```
ref class CDText;
```

C#

```
public class CDText;
```

File

IsoSDKBurnerNet.h

Description

This class will used to manage CD-Text information. The FoxBurner SDK can reaad CD-Text of the disk and a give track index. A instance of this class will created on call of the method ReadCDText.

1.1.1.8.1 CDText Members

The following tables list the members exposed by CDText.

Public Methods

	Name	Description
	GetDiskTagString (see page 88)	This function will get a specified CD-Text disk item from the created CD-Text handle of a audio disk.
	GetTrackTagString (see page 88)	This function will get a specified CD-Text track item from the created CD-Text handle of a audio disk.

1.1.1.8.2 CDText Methods

The methods of the CDText class are listed here.

Public Methods

	Name	Description
	GetDiskTagString (see page 88)	This function will get a specified CD-Text disk item from the created CD-Text handle of a audio disk.
	GetTrackTagString (see page 88)	This function will get a specified CD-Text track item from the created CD-Text handle of a audio disk.

1.1.1.8.2.1 CDText::GetDiskTagString Method

C++

```
String ^ GetDiskTagString(
    CDTextContentItem CDTCI
);
```

C#

```
public String GetDiskTagString(
    CDTextContentItem CDTCI
);
```

Description

This function will get a specified CD-Text disk item from the created CD-Text handle of a audio disk.

1.1.1.8.2.2 CDText::GetTrackTagString Method

C++

```
String ^ GetTrackTagString(
    int nTrackNumber,
    CDTextContentItem CDTCI
);
```

C#

```
public String GetTrackTagString(
    int nTrackNumber,
```

```
    CDTextContentItem CDTCI
);
```

Description

This function will get a specified CD-Text track item from the created CD-Text handle of a audio disk.

1.1.1.9 CompareFilesForArrangementEventArgs Class

Class Hierarchy

C++

```
ref class CompareFilesForArrangementEventArgs : public EventArgs;
```

C#

```
public class CompareFilesForArrangementEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the CompareFilesForArrangement event.

1.1.1.9.1 CompareFilesForArrangementEventArgs Members

The following tables list the members exposed by CompareFilesForArrangementEventArgs.

Public Data Members

	Name	Description
	m_file1 (see page 89)	File 1 to compare. A pointer to a FileEntry (see page 185) structure.
	m_file2 (see page 90)	File 2 to compare. A pointer to a FileEntry (see page 185) structure.

1.1.1.9.2 CompareFilesForArrangementEventArgs Data Members

The data members of the CompareFilesForArrangementEventArgs class are listed here.

Public Data Members

	Name	Description
	m_file1 (see page 89)	File 1 to compare. A pointer to a FileEntry (see page 185) structure.
	m_file2 (see page 90)	File 2 to compare. A pointer to a FileEntry (see page 185) structure.

1.1.1.9.2.1 CompareFilesForArrangementEventArgs::m_file1 Data Member

C++

```
FileEntry ^ m_file1;
```

C#

```
public FileEntry m_file1;
```

Description

File 1 to compare. A pointer to a FileEntry (see page 185) structure.

1.1.1.9.2.2 CompareFilesForArrangementEventArgs::m_file2 Data Member

C++

```
FileEntry ^ m_file2;
```

C#

```
public FileEntry m_file2;
```

Description

File 2 to compare. A pointer to a FileEntry (see page 185) structure.

1.1.1.10 CompressEncryptOptions Class

Class Hierarchy

IsoSDK::CompressEncryptOptions

C++

```
ref class CompressEncryptOptions;
```

C#

```
public class CompressEncryptOptions;
```

File

Options.h

Description

This class will used to manage the CompressEncryptOptions information. A instance of this class will created on call of the Property CompressEncryptOptions.

1.1.1.10.1 CompressEncryptOptions Members

The following tables list the members exposed by CompressEncryptOptions.

Public Properties

	Name	Description
	Compression (see page 91)	This parameter will enable/disable the compression function. You can enable it with true or disable it with false.
	CompressionLevel (see page 91)	The range of the compression level. The compression level range is 0 up to 9. 0 is fast compression and 9 is slow but more effective compression. Invalid values are changed to the nearest values to avoid damaged disks.
	Encryption (see page 91)	This parameter will enable/disable the encryption function. You can enable it with true or disable it with false.
	Password (see page 91)	This is the password string to decrypt the encryption later. This value is only valid upon setting. If you call to get this string it is empty due to security issues.

1.1.1.10.2 CompressEncryptOptions Properties

The properties of the CompressEncryptOptions class are listed here.

Public Properties

	Name	Description
	Compression (see page 91)	This parameter will enable/disable the compression function. You can enable it with true or disable it with false.

	CompressionLevel (see page 91)	The range of the compression level. The compression level range is 0 up to 9. 0 is fast compression and 9 is slow but more effective compression. Invalid values are changed to the nearest values to avoid damaged disks.
	Encryption (see page 91)	This parameter will enable/disable the encryption function. You can enable it with true or disable it with false.
	Password (see page 91)	This is the password string to decrypt the encryption later. This value is only valid upon setting. If you call to get this string it is empty due to security issues.

1.1.1.10.2.1 CompressEncryptOptions::Compression Property

C++

```
property bool Compression;
```

C#

```
public bool Compression;
```

Description

This parameter will enable/disable the compression function. You can enable it with true or disable it with false.

1.1.1.10.2.2 CompressEncryptOptions::CompressionLevel Property

C++

```
property int CompressionLevel;
```

C#

```
public int CompressionLevel;
```

Description

The range of the compression level. The compression level range is 0 up to 9. 0 is fast compression and 9 is slow but more effective compression. Invalid values are changed to the nearest values to avoid damaged disks.

1.1.1.10.2.3 CompressEncryptOptions::Encryption Property

C++

```
property bool Encryption;
```

C#

```
public bool Encryption;
```

Description

This parameter will enable/disable the encryption function. You can enable it with true or disable it with false.

1.1.1.10.2.4 CompressEncryptOptions::Password Property

C++

```
property String ^ Password;
```

C#

```
public String Password;
```

Description

This is the password string to decrypt the encryption later. This value is only valid upon setting. If you call to get this string it is empty due to security issues.

1.1.1.11 CreateDirEventArgs Class

Class Hierarchy

C++

```
ref class CreateDirEventArgs : public EventArgs;
```

C#

```
public class CreateDirEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the CreateDir event.

1.1.1.11.1 CreateDirEventArgs Members

The following tables list the members exposed by CreateDirEventArgs.

Public Data Members

	Name	Description
	m_strFullPath (see page 92)	A string with the original name of the created directory.
	m_strISOName (see page 93)	A string with the ISO9660 name of the created directory.
	m_strJolietName (see page 93)	A string with the Joliet name of the created directory.
	m_strUDFName (see page 93)	A string with the UDF name of the created directory.

1.1.1.11.2 CreateDirEventArgs Data Members

The data members of the CreateDirEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strFullPath (see page 92)	A string with the original name of the created directory.
	m_strISOName (see page 93)	A string with the ISO9660 name of the created directory.
	m_strJolietName (see page 93)	A string with the Joliet name of the created directory.
	m_strUDFName (see page 93)	A string with the UDF name of the created directory.

1.1.1.11.2.1 CreateDirEventArgs::m_strFullPath Data Member

C++

```
String ^ m_strFullPath;
```

C#

```
public String m_strFullPath;
```

Description

A string with the original name of the created directory.

1.1.1.11.2.2 CreateDirEventArgs::m_strISOName Data Member

C++

```
String ^ m_strISOName;
```

C#

```
public String m_strISOName;
```

Description

A string with the ISO9660 name of the created directory.

1.1.1.11.2.3 CreateDirEventArgs::m_strJolietName Data Member

C++

```
String ^ m_strJolietName;
```

C#

```
public String m_strJolietName;
```

Description

A string with the Joliet name of the created directory.

1.1.1.11.2.4 CreateDirEventArgs::m_strUDFName Data Member

C++

```
String ^ m_strUDFName;
```

C#

```
public String m_strUDFName;
```

Description

A string with the UDF name of the created directory.

1.1.1.12 DiskDirectory Class

Class Hierarchy

IsoSDK::DiskDirectory

C++

```
ref class DiskDirectory;
```

C#

```
public class DiskDirectory;
```

File

IsoSDKBurnerNet.h

Description

This class will used to manage a directroy of a DiskSession (see page 94) object. A instance of this class will created on call of the method OpenDirectory.

1.1.1.12.1 DiskDirectory Members

The following tables list the members exposed by DiskDirectory.

Public Properties

	Name	Description
R	Files (see page 94)	Provides indexed access to the directory contents. Each entry corresponds to a child file or directory. To get know the number of entries use FilesCount (see page 94) property.
R	FilesCount (see page 94)	Gets number of child entries of the directory.

1.1.1.12.2 DiskDirectory Properties

The properties of the DiskDirectory class are listed here.

Public Properties

	Name	Description
R	Files (see page 94)	Provides indexed access to the directory contents. Each entry corresponds to a child file or directory. To get know the number of entries use FilesCount (see page 94) property.
R	FilesCount (see page 94)	Gets number of child entries of the directory.

1.1.1.12.2.1 DiskDirectory::Files Property

C++

```
property FileEntry ^ Files[int];
```

C#

```
public FileEntry Files;
```

Description

Provides indexed access to the directory contents. Each entry corresponds to a child file or directory. To get know the number of entries use FilesCount (see page 94) property.

1.1.1.12.2.2 DiskDirectory::FilesCount Property

C++

```
property int FilesCount;
```

C#

```
public int FilesCount;
```

Description

Gets number of child entries of the directory.

1.1.1.13 DiskSession Class

Class Hierarchy

IsoSDK::DiskSession

C++

```
ref class DiskSession;
```

C#

```
public class DiskSession;
```

File

IsoSDKBurnerNet.h

Description

This class will used to manage a session of a Burner (see page 13) object. A instance of this class will created on call of the method OpenDiskSession.

1.1.1.13.1 DiskSession Members

The following tables list the members exposed by DiskSession.

Public Methods

	Name	Description
	GetBootVolumeInformation (see page 95)	Use this function to receive information about the boot file system on a existing disk or image.
	GetFileAllocationTable (see page 96)	Use this function to receive information of the file allocation table and the file extents on the disk according to the given file.
	GetISOVolumeInformation (see page 96)	Use this function to receive information about the ISO file system on a existing disk or image.
	GetUDFVolumeInformation (see page 96)	Use this function to receive information about the UDF file system on a existing disk or image.
	ImportFile (see page 96)	This function imports a file of a session to your project.
	ImportFileEx (see page 97)	This function imports a compressed and/or encrypted file of a session to your project.
	OpenDirectory (see page 97)	Creates a DiskDirectory (see page 93) object of a DiskSession (see page 94).
	VerifyFile (see page 98)	This function verify a file of the medium against whether it is is readable without any error.

1.1.1.13.2 DiskSession Methods

The methods of the DiskSession class are listed here.

Public Methods

	Name	Description
	GetBootVolumeInformation (see page 95)	Use this function to receive information about the boot file system on a existing disk or image.
	GetFileAllocationTable (see page 96)	Use this function to receive information of the file allocation table and the file extents on the disk according to the given file.
	GetISOVolumeInformation (see page 96)	Use this function to receive information about the ISO file system on a existing disk or image.
	GetUDFVolumeInformation (see page 96)	Use this function to receive information about the UDF file system on a existing disk or image.
	ImportFile (see page 96)	This function imports a file of a session to your project.
	ImportFileEx (see page 97)	This function imports a compressed and/or encrypted file of a session to your project.
	OpenDirectory (see page 97)	Creates a DiskDirectory (see page 93) object of a DiskSession (see page 94).
	VerifyFile (see page 98)	This function verify a file of the medium against whether it is is readable without any error.

1.1.1.13.2.1 DiskSession::GetBootVolumeInformation Method

C++

```
BootVolumeInfo ^ GetBootVolumeInformation();
```

C#

```
public BootVolumeInfo GetBootVolumeInformation();
```

Returns

A pointer to a BootVolumeInfo (see page 144) structure.

Description

Use this function to receive information about the boot file system on a existing disk or image.

1.1.1.13.2.2 DiskSession::GetFileAllocationTable Method

C++

```
FileAllocationTable ^ GetFileAllocationTable(
    String ^ filePath
);
```

C#

```
public FileAllocationTable GetFileAllocationTable(
    ref String filePath
);
```

Returns

A pointer to a FileAllocationTable (see page 183) structure.

Description

Use this function to receive information of the file allocation table and the file extents on the disk according to the given file.

1.1.1.13.2.3 DiskSession::GetISOVolumeInformation Method

C++

```
ISOVolumeInfo ^ GetISOVolumeInformation();
```

C#

```
public ISOVolumeInfo GetISOVolumeInformation();
```

Returns

A pointer to a ISOVolumeInfo (see page 189) structure.

Description

Use this function to receive information about the ISO file system on a existing disk or image.

1.1.1.13.2.4 DiskSession::GetUDFVolumeInformation Method

C++

```
UDFVolumeInfo ^ GetUDFVolumeInformation();
```

C#

```
public UDFVolumeInfo GetUDFVolumeInformation();
```

Returns

A pointer to a UDFVolumeInfo (see page 206) structure.

Description

Use this function to receive information about the UDF file system on a existing disk or image.

1.1.1.13.2.5 DiskSession::ImportFile Method

C++

```
bool ImportFile(
    String ^ SourcePath,
    String ^ DestFolderPath
);
```

C#

```
public bool ImportFile(
    ref String SourcePath,
    ref String DestFolderPath
);
```

Parameters

Parameters	Description
String ^ SourcePath	Handle to the string that names the imported directory.
String ^ DestFolderPath	Handle to the string that names the destination directory of the project.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function imports a file of a session to your project.

1.1.1.13.2.6 DiskSession::ImportFileEx Method

C++

```
bool ImportFileEx(
    String^ SourcePath,
    String^ DestFolderPath,
    String^ Password
);
```

C#

```
public bool ImportFileEx(
    ref String SourcePath,
    ref String DestFolderPath,
    ref String Password
);
```

Parameters

Parameters	Description
String^ SourcePath	Handle to the string that names the imported directory.
String^ DestFolderPath	Handle to the string that names the destination directory of the project.
String^ Password	Handle to the string that holds the password to encrypt.

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function imports a compressed and/or encrypted file of a session to your project.

1.1.1.13.2.7 DiskSession::OpenDirectory Method

C++

```
DiskDirectory ^ OpenDirectory(
    String ^ Path
);
```

C#

```
public DiskDirectory OpenDirectory(
    ref String Path
```

```
);
```

Parameters

Parameters	Description
String ^ Path	A string that names the directory to be opened.

Returns

This function returns a pointer to a class DiskDirectory (see page 93) for the given path.

Description

Creates a DiskDirectory (see page 93) object of a DiskSession (see page 94).

1.1.1.13.2.8 DiskSession::VerifyFile Method

C++

```
bool VerifyFile(
    String ^ SourcePath
);
```

C#

```
public bool VerifyFile(
    ref String SourcePath
);
```

Parameters

Parameters	Description
String ^ SourcePath	The path to the file on the medium. Example "\data\testfile.txt"

Returns

The function returns true upon successful execution. In case of an error the error code will be returned (see ErrorCode (see page 155)).

Description

This function verify a file of the medium against whether it is is readable without any error.

1.1.1.14 DVDVideoOptions Class

Class Hierarchy

IsoSDK::DVDVideoOptions

C++

```
ref class DVDVideoOptions;
```

C#

```
public class DVDVideoOptions;
```

File

Options.h

Description

This class will used to manage the DVDVideoOptions information. A instance of this class will created on call of the Property DVDVideoOptions.

1.1.1.14.1 DVDVideoOptions Members

The following tables list the members exposed by DVDVideoOptions.

Public Properties

	Name	Description
	ForceUppercase (see page 99)	Will make all files on the disk uppercase.
	Padding (see page 99)	IFO 32K Padding. Some DVD authoring applications use this option to set a fixed place between IFO file and next file. If you have a VideoDVD image that was authored with this option you have to set this flag to TRUE to avoid playback problems.

1.1.1.14.2 DVDVideoOptions Properties

The properties of the DVDVideoOptions class are listed here.

Public Properties

	Name	Description
	ForceUppercase (see page 99)	Will make all files on the disk uppercase.
	Padding (see page 99)	IFO 32K Padding. Some DVD authoring applications use this option to set a fixed place between IFO file and next file. If you have a VideoDVD image that was authored with this option you have to set this flag to TRUE to avoid playback problems.

1.1.1.14.2.1 DVDVideoOptions::ForceUppercase Property

C++

```
property bool ForceUppercase;
```

C#

```
public bool ForceUppercase;
```

Description

Will make all files on the disk uppercase.

1.1.1.14.2.2 DVDVideoOptions::Padding Property

C++

```
property bool Padding;
```

C#

```
public bool Padding;
```

Description

IFO 32K Padding. Some DVD authoring applications use this option to set a fixed place between IFO file and next file. If you have a VideoDVD image that was authored with this option you have to set this flag to TRUE to avoid playback problems.

1.1.1.15 EraseDoneEventArgs Class

Class Hierarchy

C++

```
ref class EraseDoneEventArgs : public EventArgs;
```

C#

```
public class EraseDoneEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the EraseDone event.

1.1.1.15.1 EraseDoneEventArgs Members

The following tables list the members exposed by EraseDoneEventArgs.

Public Data Members

	Name	Description
	m_strError (see page 100)	A string with the deletion error. If no error occurred, an empty string will be returned.

1.1.1.15.2 EraseDoneEventArgs Data Members

The data members of the EraseDoneEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strError (see page 100)	A string with the deletion error. If no error occurred, an empty string will be returned.

1.1.1.15.2.1 EraseDoneEventArgs::m_strError Data Member

C++

```
String ^ m_strError;
```

C#

```
public String m_strError;
```

Description

A string with the deletion error. If no error occurred, an empty string will be returned.

1.1.1.16 ExtendedDeviceCapabilities Class

Class Hierarchy

IsoSDK::ExtendedDeviceCapabilities

C++

```
ref class ExtendedDeviceCapabilities;
```

C#

```
public class ExtendedDeviceCapabilities;
```

File

IsoSDKBurnerNet.h

Description

This class will used to manage the extended device capabilities information. A instance of this class will created on call of the method GetDeviceInformationEx.

1.1.1.16.1 ExtendedDeviceCapabilities Members

The following tables list the members exposed by ExtendedDeviceCapabilities.

Public Methods

	Name	Description
	AnalyseCapability (see page 101)	With this method you can analyze all possible capabilities. You can analyze as many capabilities and as many times as needed.

1.1.1.16.2 ExtendedDeviceCapabilities Methods

The methods of the ExtendedDeviceCapabilities class are listed here.

Public Methods

	Name	Description
	AnalyseCapability (see page 101)	With this method you can analyze all possible capabilities. You can analyze as many capabilities and as many times as needed.

1.1.1.16.2.1 ExtendedDeviceCapabilities::AnalyseCapability Method

C++

```
bool AnalyseCapability(
    Capabilities OneCapability
);
```

C#

```
public bool AnalyseCapability(
    Capabilities OneCapability
);
```

Parameters

Parameters	Description
Capabilities OneCapability	A value of the enumeration Capabilities (see page 146).

Returns

The method will return true if the queried capabilitie is supported. Else it returns false.

Description

With this method you can analyze all possible capabilities. You can analyze as many capabilities and as many times as needed.

1.1.1.17 FileDateTime Structure

C++

```
ref struct FileDateTime {
  int8 Year;
  int8 Month;
  int8 Day;
  int8 Hour;
  int8 Minute;
  int8 Second;
};
```

C#

```
public struct FileDateTime {
  public int8 Year;
  public int8 Month;
```

```
    public int8 Day;
    public int8 Hour;
    public int8 Minute;
    public int8 Second;
  }
```

File

IsoSDKBurnerNet.h

Description

A structure that contains the information about the time & date of the specific file.

1.1.1.17.1 FileDateTime Members

The following tables list the members exposed by FileDateTime.

Public Data Members

	Name	Description
	Day (see page 102)	Day of month (1.....31)
	Hour (see page 103)	Hours (0...23)
	Minute (see page 103)	Minutes (0...59)
	Month (see page 103)	Month (1..12)
	Second (see page 103)	Seconds (0...59)
	Year (see page 103)	Years since 1900

Public Methods

	Name	Description
	ToDateTime (see page 104)	This is ToDateTime, a member of class FileDateTime.

1.1.1.17.2 FileDateTime Data Members

The data members of the FileDateTime class are listed here.

Public Data Members

	Name	Description
	Day (see page 102)	Day of month (1.....31)
	Hour (see page 103)	Hours (0...23)
	Minute (see page 103)	Minutes (0...59)
	Month (see page 103)	Month (1..12)
	Second (see page 103)	Seconds (0...59)
	Year (see page 103)	Years since 1900

1.1.1.17.2.1 FileDateTime::Day Data Member

C++

```
int8 Day;
```

C#

```
public int8 Day;
```

Description

Day of month (1.....31)

1.1.1.17.2.2 FileDateTime::Hour Data Member

C++

```
int8 Hour;
```

C#

```
public int8 Hour;
```

Description

Hours (0...23)

1.1.1.17.2.3 FileDateTime::Minute Data Member

C++

```
int8 Minute;
```

C#

```
public int8 Minute;
```

Description

Minutes (0...59)

1.1.1.17.2.4 FileDateTime::Month Data Member

C++

```
int8 Month;
```

C#

```
public int8 Month;
```

Description

Month (1..12)

1.1.1.17.2.5 FileDateTime::Second Data Member

C++

```
int8 Second;
```

C#

```
public int8 Second;
```

Description

Seconds (0...59)

1.1.1.17.2.6 FileDateTime::Year Data Member

C++

```
int8 Year;
```

C#

```
public int8 Year;
```

Description

Years since 1900

1.1.1.17.3 FileDateTime Methods

The methods of the FileDateTime class are listed here.

Public Methods

	Name	Description
	ToDateTime (see page 104)	This is ToDateTime, a member of class FileDateTime.

1.1.1.17.3.1 FileDateTime::ToDateTime Method

C++

```
DateTime ToDateTime();
```

C#

```
public DateTime ToDateTime();
```

Description

This is ToDateTime, a member of class FileDateTime.

1.1.1.18 FileDateTimeEx Class new

Class Hierarchy

IsoSDK::FileDateTimeEx

C++

```
ref class FileDateTimeEx;
```

C#

```
public class FileDateTimeEx;
```

File

Options.h

Description

This class is used to manage the custom date / time values for files, directory and file system entries. With this values you can overwrite the common date / time values of the files.

You can use the system time or own time values. This date / time value is set global but you can overwrite each file value with SetFileTimes method.

This settings will not changed old added files / directories only those that get added after calling this. Will get reset with define a new project.

1.1.1.18.1 FileDateTimeEx Members

The following tables list the members exposed by FileDateTimeEx.

Public Properties

	Name	Description
new	CreationDateTime (see page 105)	The CreationDateTime value.
new	LastAccessDateTime (see page 105)	The LastAccessDateTime value.
new	ModificationDateTime (see page 106)	The ModificationDateTime value.

new	UseCreationDateTime (see page 106)	Switch to enable the CreationDateTime (see page 105) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.
new	UseCustomTimes (see page 106)	The global switch to enable the custom date / time information for files, directories and file systems. If the FS only allow one date, the Modification date / time info is used.
new	UseLastAccessDateTime (see page 106)	Switch to enable the LastAccessDateTime (see page 105) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.
new	UseModificationDateTime (see page 106)	Switch to enable the ModificationDateTime (see page 106) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.

1.1.1.18.2 FileDateTimeEx Properties

The properties of the FileDateTimeEx class are listed here.

Public Properties

	Name	Description
new	CreationDateTime (see page 105)	The CreationDateTime value.
new	LastAccessDateTime (see page 105)	The LastAccessDateTime value.
new	ModificationDateTime (see page 106)	The ModificationDateTime value.
new	UseCreationDateTime (see page 106)	Switch to enable the CreationDateTime (see page 105) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.
new	UseCustomTimes (see page 106)	The global switch to enable the custom date / time information for files, directories and file systems. If the FS only allow one date, the Modification date / time info is used.
new	UseLastAccessDateTime (see page 106)	Switch to enable the LastAccessDateTime (see page 105) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.
new	UseModificationDateTime (see page 106)	Switch to enable the ModificationDateTime (see page 106) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.

1.1.1.18.2.1 FileDateTimeEx::CreationDateTime Property new

C++

```
property FileDateTime^ CreationDateTime;
```

C#

```
public FileDateTime CreationDateTime;
```

Description

The CreationDateTime value.

1.1.1.18.2.2 FileDateTimeEx::LastAccessDateTime Property new

C++

```
property FileDateTime^ LastAccessDateTime;
```

C#

```
public FileDateTime LastAccessDateTime;
```

Description

The LastAccessDateTime value.

1.1.1.18.2.3 FileDateTimeEx::ModificationDateTime Property new

C++

```
property FileDateTime^ ModificationDateTime;
```

C#

```
public FileDateTime ModificationDateTime;
```

Description

The ModificationDateTime value.

1.1.1.18.2.4 FileDateTimeEx::UseCreationDateTime Property new

C++

```
property bool UseCreationDateTime;
```

C#

```
public bool UseCreationDateTime;
```

Description

Switch to enable the CreationDateTime (see page 105) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.

1.1.1.18.2.5 FileDateTimeEx::UseCustomTimes Property new

C++

```
property bool UseCustomTimes;
```

C#

```
public bool UseCustomTimes;
```

Description

The global switch to enable the custom date / time information for files, directories and file systems. If the FS only allow one date, the Modification date / time info is used.

1.1.1.18.2.6 FileDateTimeEx::UseLastAccessDateTime Property new

C++

```
property bool UseLastAccessDateTime;
```

C#

```
public bool UseLastAccessDateTime;
```

Description

Switch to enable the LastAccessDateTime (see page 105) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.

1.1.1.18.2.7 FileDateTimeEx::UseModificationDateTime Property new

C++

```
property bool UseModificationDateTime;
```

C#

```
public bool UseModificationDateTime;
```

Description

Switch to enable the ModificationDateTime (see page 106) value. Only enabled if the UseCustomTimes (see page 106) switch is also set to true.

1.1.1.19 GeneralOptions Class

Class Hierarchy

IsoSDK::GeneralOptions

C++

```
ref class GeneralOptions;
```

C#

```
public class GeneralOptions;
```

File

Options.h

Description

This class will used to manage the GeneralOptions information. A instance of this class will created on call of the Property Options.

1 = BS_BURNER_DATA	2 = BS_BURNER_UDFDVD	3 = BS_BURNER_ISOUDF
4 = BS_BURNER_BLURAY	5 = BS_BURNER_AUDIO	6 = BS_BURNER_MIXEDMODE
7 = BS_BURNER_VIDEODVD	8 = BS_BURNER_VCD	9 = BS_BURNER_SVCD
10 = BS_BURNER_CUE	11 = BS_BURNER_RAW	

	1	2	3	4	5	6	7	8	9	10	11
Label	x	x	x	x		x	x	x	x		
Write Method *1	x	x	x		x	x	x	x	x		
Joliet (see page 111)	x		x			x					
Boot	x		x								
Finalize	x	x	x	x							
Test Burning *1	x	x	x	x	x	x	x	x	x	x	x
OPC	x	x	x	x	x	x	x	x	x	x	x
Verify	x	x	x	x		x *2	x				
Eject	x	x	x	x	x	x	x	x	x	x	x
UnderrunProtection (see page 112)	x	x	x	x	x	x	x	x	x	x	x
UDF		x	x	x							
IsoEx	x		x			x	x				
Compression	x										
Encryption	x										
Pad Data Tracks	x	x	x	x		x					

*1 = Only valid for disc type CD

*2 = Only the ISO part will get verified

1.1.1.19.1 GeneralOptions Members

The following tables list the members exposed by GeneralOptions.

Public Properties

	Name	Description
	AutoErase (see page 110)	This argument sets the auto erase function for CD/DVD/BluRay ReWriteable. If this option is set to true the IsoSDK (see page 1) will erase a RW before the new burn job started. Please note that this option use the "Quick erase" function.
	Bootable (see page 110)	Get/set the bootable switch. Pass true to activate it and pass false to deactivate it. This option is only available for data projects. The BootOptions (see page 10) have to be set for this switch.
	BootImage (see page 110)	Get/Set the file name of the image (with path information). This represent the boot image you use with the IsoSDK (see page 1).
	CacheSize (see page 110)	Sets the size of the burning buffer to be used (in bytes; default: 4 * 1024 * 1024 [4 MB]) We recommend to use a range between 4 MB and 32 MB. 4 MB = CD 16 MB = DVD 32 MB = BD
	Copies (see page 111)	Get/Set the copies to burn. Only available in GUI mode.
	EjectAfterBurn (see page 111)	Get/Set whether the medium will be ejected after the burning process. true ejects the medium, and false prevents this.
	FinalizeDisk (see page 111)	Get/Set whether the medium should be finalized. true finalizes the medium, whereas false burns a multi-session medium).
	Joliet (see page 111)	Get/Set whether to use the Joliet file system, pass true to activate it and pass false to deactivate it. This option is only available for data projects.
	PadDataTracks (see page 111)	This option will pad the data track of a disk to the min. size of 4 seconds. This is important for Data, Multisession and Mixed Mode disks if your work with very small sizes.
	PerformOPC (see page 112)	**Optimum Power Calibration.** DVD writing devices perform a test write and read in an area inside of the lead-in, in order to determine the best laser power for recording. This allows the device to adjust to each disc, which may vary slightly from different manufacturers, or for other reasons. Get/Set whether you want to perform OPC or not. Pass true to activate it and pass false to de-activate it.
	RockRidge (see page 112)	Get/Set whether you want to write the RockRidge Extension. Pass true to activate it and pass false to de-activate it.
	TestBurn (see page 112)	Get/Set the simulation option. The value "true" activates the burning simulation and false skips the simulation. The IsoSDK (see page 1) will make a dry run with this option. To use the final write to the disc you have to set this value to false.
	UnderrunProtection (see page 112)	Get/Set whether the buffer-underrun-protection will be used.
	VerifyAfterBurn (see page 112)	Get/Set the verify after burn. This option is only available in Data CD/DVD ISO and UDF.
	VolumeLabel (see page 113)	Get/set the volume name of the medium. This name will be displayed e.g. in the Windows Explorer. This string may neither contain high characters nor spaces. Max. 128 character long.
	WriteMethod (see page 113)	Get/Set the write method. The property will be filled according to the WriteMethod enumeration. You can set this only on CD media. For DVD and BD the IsoSDK (see page 1) will set the method itself.

1.1.1.19.2 GeneralOptions Properties

The properties of the GeneralOptions class are listed here.

Public Properties

	Name	Description
	AutoErase (see page 110)	This argument sets the auto erase function for CD/DVD/BluRay ReWriteable. If this option is set to true the IsoSDK (see page 1) will erase a RW before the new burn job started. Please note that this option use the "Quick erase" function.
	Bootable (see page 110)	Get/set the bootable switch. Pass true to activate it and pass false to deactivate it. This option is only available for data projects. The BootOptions (see page 10) have to be set for this switch.
	BootImage (see page 110)	Get/Set the file name of the image (with path information). This represent the boot image you use with the IsoSDK (see page 1).
	CacheSize (see page 110)	Sets the size of the burning buffer to be used (in bytes; default: 4 * 1024 * 1024 [4 MB]) We recommend to use a range between 4 MB and 32 MB. 4 MB = CD 16 MB = DVD 32 MB = BD
	Copies (see page 111)	Get/Set the copies to burn. Only available in GUI mode.
	EjectAfterBurn (see page 111)	Get/Set whether the medium will be ejected after the burning process. true ejects the medium, and false prevents this.
	FinalizeDisk (see page 111)	Get/Set whether the medium should be finalized. true finalizes the medium, whereas false burns a multi-session medium).
	Joliet (see page 111)	Get/Set whether to use the Joliet file system, pass true to activate it and pass false to deactivate it. This option is only available for data projects.
	PadDataTracks (see page 111)	This option will pad the data track of a disk to the min. size of 4 seconds. This is important for Data, Multisession and Mixed Mode disks if your work with very small sizes.
	PerformOPC (see page 112)	**Optimum Power Calibration.** DVD writing devices perform a test write and read in an area inside of the lead-in, in order to determine the best laser power for recording. This allows the device to adjust to each disc, which may vary slightly from different manufacturers, or for other reasons. Get/Set whether you want to perform OPC or not. Pass true to activate it and pass false to de-activate it.
	RockRidge (see page 112)	Get/Set whether you want to write the RockRidge Extension. Pass true to activate it and pass false to de-activate it.
	TestBurn (see page 112)	Get/Set the simulation option. The value "true" activates the burning simulation and false skips the simulation. The IsoSDK (see page 1) will make a dry run with this option. To use the final write to the disc you have to set this value to false.
	UnderrunProtection (see page 112)	Get/Set whether the buffer-underrun-protection will be used.
	VerifyAfterBurn (see page 112)	Get/Set the verify after burn. This option is only available in Data CD/DVD ISO and UDF.
	VolumeLabel (see page 113)	Get/set the volume name of the medium. This name will be displayed e.g. in the Windows Explorer. This string may neither contain high characters nor spaces. Max. 128 character long.
	WriteMethod (see page 113)	Get/Set the write method. The property will be filled according to the WriteMethod enumeration. You can set this only on CD media. For DVD and BD the IsoSDK (see page 1) will set the method itself.

1.1.1.19.2.1 GeneralOptions::AutoErase Property

C++

```
property bool AutoErase;
```

C#

```
public bool AutoErase;
```

Description

This argument sets the auto erase function for CD/DVD/BluRay ReWriteable. If this option is set to true the IsoSDK (see page 1) will erase a RW before the new burn job started. Please note that this option use the "Quick erase" function.

1.1.1.19.2.2 GeneralOptions::Bootable Property

C++

```
property bool Bootable;
```

C#

```
public bool Bootable;
```

Description

Get/set the bootable switch. Pass true to activate it and pass false to deactivate it. This option is only available for data projects.

The BootOptions (see page 10) have to be set for this switch.

1.1.1.19.2.3 GeneralOptions::BootImage Property

C++

```
property String ^ BootImage;
```

C#

```
public String BootImage;
```

Description

Get/Set the file name of the image (with path information).

This represent the boot image you use with the IsoSDK (see page 1).

1.1.1.19.2.4 GeneralOptions::CacheSize Property

C++

```
property int CacheSize;
```

C#

```
public int CacheSize;
```

Description

Sets the size of the burning buffer to be used (in bytes; default: 4 * 1024 * 1024 [4 MB])

We recommend to use a range between 4 MB and 32 MB.

4 MB = CD

16 MB = DVD

32 MB = BD

1.1.1.19.2.5 GeneralOptions::Copies Property

C++

```
property int Copies;
```

C#

```
public int Copies;
```

Description

Get/Set the copies to burn. Only available in GUI mode.

1.1.1.19.2.6 GeneralOptions::EjectAfterBurn Property

C++

```
property bool EjectAfterBurn;
```

C#

```
public bool EjectAfterBurn;
```

Description

Get/Set whether the medium will be ejected after the burning process. true ejects the medium, and false prevents this.

1.1.1.19.2.7 GeneralOptions::FinalizeDisk Property

C++

```
property bool FinalizeDisk;
```

C#

```
public bool FinalizeDisk;
```

Description

Get/Set whether the medium should be finalized. true finalizes the medium, whereas false burns a multi-session medium).

1.1.1.19.2.8 GeneralOptions::Joliet Property

C++

```
property bool Joliet;
```

C#

```
public bool Joliet;
```

Description

Get/Set whether to use the Joliet file system, pass true to activate it and pass false to deactivate it. This option is only available for data projects.

1.1.1.19.2.9 GeneralOptions::PadDataTracks Property

C++

```
property bool PadDataTracks;
```

C#

```
public bool PadDataTracks;
```

Description

This option will pad the data track of a disk to the min. size of 4 seconds. This is important for Data, Multisession and Mixed Mode disks if your work with very small sizes.

1.1.1.19.2.10 GeneralOptions::PerformOPC Property

C++

```
property bool PerformOPC;
```

C#

```
public bool PerformOPC;
```

Description

Optimum Power Calibration.

DVD writing devices perform a test write and read in an area inside of the lead-in, in order to determine the best laser power for recording. This allows the device to adjust to each disc, which may vary slightly from different manufacturers, or for other reasons.

Get/Set whether you want to perform OPC or not. Pass true to activate it and pass false to de-activate it.

1.1.1.19.2.11 GeneralOptions::RockRidge Property

C++

```
property bool RockRidge;
```

C#

```
public bool RockRidge;
```

Description

Get/Set whether you want to write the RockRidge Extension. Pass true to activate it and pass false to de-activate it.

1.1.1.19.2.12 GeneralOptions::TestBurn Property

C++

```
property bool TestBurn;
```

C#

```
public bool TestBurn;
```

Description

Get/Set the simulation option. The value "true" activates the burning simulation and false skips the simulation.

The IsoSDK (see page 1) will make a dry run with this option. To use the final write to the disc you have to set this value to false.

1.1.1.19.2.13 GeneralOptions::UnderrunProtection Property

C++

```
property bool UnderrunProtection;
```

C#

```
public bool UnderrunProtection;
```

Description

Get/Set whether the buffer-underrun-protection will be used.

1.1.1.19.2.14 GeneralOptions::VerifyAfterBurn Property

C++

```
property bool VerifyAfterBurn;
```

C#

```
public bool VerifyAfterBurn;
```

Description

Get/Set the verify after burn. This option is only available in Data CD/DVD ISO and UDF.

1.1.1.19.2.15 GeneralOptions::VolumeLabel Property

C++

```
property String ^ VolumeLabel;
```

C#

```
public String VolumeLabel;
```

Description

Get/set the volume name of the medium. This name will be displayed e.g. in the Windows Explorer. This string may neither contain high characters nor spaces. Max. 128 character long.

1.1.1.19.2.16 GeneralOptions::WriteMethod Property

C++

```
property IsoSDK::WriteMethod WriteMethod;
```

C#

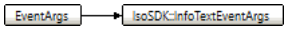
```
public IsoSDK::WriteMethod WriteMethod;
```

Description

Get/Set the write method. The property will be filled according to the WriteMethod enumeration. You can set this only on CD media. For DVD and BD the IsoSDK (see page 1) will set the method itself.

1.1.1.20 InfoTextEventArgs Class

Class Hierarchy



C++

```
ref class InfoTextEventArgs : public EventArgs;
```

C#

```
public class InfoTextEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the InfoText event.

1.1.1.20.1 InfoTextEventArgs Members

The following tables list the members exposed by InfoTextEventArgs.

Public Data Members

	Name	Description
	m_nLevel (see page 114)	The type of the information level according to the InfoLevel (see page 187) enumeration.

	m_strInfoText (see page 114)	A string with the extended information about the current status of the IsoSDK (see page 1).

1.1.1.20.2 InfoTextEventArgs Data Members

The data members of the InfoTextEventArgs class are listed here.

Public Data Members

	Name	Description
	m_nLevel (see page 114)	The type of the information level according to the InfoLevel (see page 187) enumeration.
	m_strInfoText (see page 114)	A string with the extended information about the current status of the IsoSDK (see page 1).

1.1.1.20.2.1 InfoTextEventArgs::m_nLevel Data Member

C++

```
InfoLevel m_nLevel;
```

C#

```
public InfoLevel m_nLevel;
```

Description

The type of the information level according to the InfoLevel (see page 187) enumeration.

1.1.1.20.2.2 InfoTextEventArgs::m_strInfoText Data Member

C++

```
String ^ m_strInfoText;
```

C#

```
public String m_strInfoText;
```

Description

A string with the extended information about the current status of the IsoSDK (see page 1).

1.1.1.21 ISOExOptions Class

Class Hierarchy

IsoSDK::ISOExOptions

C++

```
ref class ISOExOptions;
```

C#

```
public class ISOExOptions;
```

File

Options.h

Description

This class will used to manage ISOExOptions information. A instance of this class will created on call of the Property ISOExOptions.

1.1.1.21.1 ISOExOptions Members

The following tables list the members exposed by ISOExOptions.

Public Properties

	Name	Description
	AddSuffix (see page 117)	Get/Set the bool value to enable ISO version number extension (;1). The ISO 9660 specification requires that each filename include a ;x version number as the suffix. A full filename in ISO 9660 has three parts: <name>.<extension>;<version> Presumably the version part was designed to allow multiple versions of the same file to coexist.
	AllowLongISO9660Names (see page 117)	If enabled (true), allows long file and directory names up to 207 ASCII characters in ISO9660 file system. If options is disabled (false), the ISO Level restrictions on file name length will be used.
	AllowLongJolietNames (see page 117)	If enabled (true), allows long file and directory names up to 103 UCS-2 characters in Joliet extension to ISO9660 file system. If options is disabled (false), the max file name length will be limited by 64 UCS-2 characters.
	AllowLowercaseNames (see page 118)	Get/Set the bool value to allow lower case letters in file names.
	AllowManyDirectories (see page 118)	Get/Set the bool value to allow path dept of more than 8 directories.
	ApplicationIdentifier (see page 118)	Get/Set the Volume Application Identifier field of the ISO/Joliet image(128 characters maximum).For detailed information see: ISO 9660
	BiblioIdentifier (see page 118)	Get/Set the Volume Bibliography File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660
	CopyrightFile (see page 118)	Get/Set the Volume Copyright File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660
	CreationDateTime (see page 119)	The creation date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	DataPreparer (see page 119)	Get/Set the Volume Data Preparer Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660
	EffectiveDateTime (see page 119)	The effective date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	ExpirationDateTime (see page 119)	The expiration date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	FileIdentifier (see page 120)	Get/Set the Volume Abstract File Identifier field of the ISO/Joliet image (35 characters maximum). For detailed information see: ISO 9660
	ISOLevel (see page 120)	Get/set the ISOLevel for ISO9660 according to the ISOLevel enumeration.
	ModificationDateTime (see page 120)	The modification date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	Publisher (see page 120)	Get/Set the Volume Publisher Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660
	SystemIdentifier (see page 120)	Get/Set the Volume System Identifier field of the ISO/Joliet image (31 characters maximum). For detailed information see: ISO 9660
	UseCreationDateTime (see page 121)	Switch to activate the CreationDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by CreationDateTime (see page 119). Default false, will use current date.
	UseEffectiveDateTime (see page 121)	Switch to activate the EffectiveDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by EffectiveDateTime (see page 119). Default false, will use current date.

	UseExpirationDateTime (see page 121)	Switch to activate the ExpirationDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by ExpirationDateTime (see page 119). Default false, will use current date.
	UseModificationDateTime (see page 121)	Switch to activate the ModificationDateTime (see page 120) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by ModificationDateTime (see page 120). Default false, will use current date.
	VolumeSet (see page 121)	Get/Set the Volume Set Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2 ISOExOptions Properties

The properties of the ISOExOptions class are listed here.

Public Properties

	Name	Description
	AddSuffix (see page 117)	Get/Set the bool value to enable ISO version number extension (;1). The ISO 9660 specification requires that each filename include a ;x version number as the suffix. A full filename in ISO 9660 has three parts: <name>.<extension>;<version> Presumably the version part was designed to allow multiple versions of the same file to coexist.
	AllowLongISO9660Names (see page 117)	If enabled (true), allows long file and directory names up to 207 ASCII characters in ISO9660 file system. If options is disabled (false), the ISO Level restrictions on file name length will be used.
	AllowLongJolietNames (see page 117)	If enabled (true), allows long file and directory names up to 103 UCS-2 characters in Joliet extension to ISO9660 file system. If options is disabled (false), the max file name length will be limited by 64 UCS-2 characters.
	AllowLowercaseNames (see page 118)	Get/Set the bool value to allow lower case letters in file names.
	AllowManyDirectories (see page 118)	Get/Set the bool value to allow path dept of more than 8 directories.
	ApplicationIdentifier (see page 118)	Get/Set the Volume Application Identifier field of the ISO/Joliet image(128 characters maximum).For detailed information see: ISO 9660
	BiblioIdentifier (see page 118)	Get/Set the Volume Bibliography File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660
	CopyrightFile (see page 118)	Get/Set the Volume Copyright File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660
	CreationDateTime (see page 119)	The creation date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	DataPreparer (see page 119)	Get/Set the Volume Data Preparer Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660
	EffectiveDateTime (see page 119)	The effective date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	ExpirationDateTime (see page 119)	The expiration date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	FileIdentifier (see page 120)	Get/Set the Volume Abstract File Identifier field of the ISO/Joliet image (35 characters maximum). For detailed information see: ISO 9660
	ISOLevel (see page 120)	Get/set the ISOLevel for ISO9660 according to the ISOLevel enumeration.
	ModificationDateTime (see page 120)	The modification date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.
	Publisher (see page 120)	Get/Set the Volume Publisher Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660

	SystemIdentifier (see page 120)	Get/Set the Volume System Identifier field of the ISO/Joliet image (31 characters maximum). For detailed information see: ISO 9660
	UseCreationDateTime (see page 121)	Switch to activate the CreationDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by CreationDateTime (see page 119). Default false, will use current date.
	UseEffectiveDateTime (see page 121)	Switch to activate the EffectiveDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by EffectiveDateTime (see page 119). Default false, will use current date.
	UseExpirationDateTime (see page 121)	Switch to activate the ExpirationDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by ExpirationDateTime (see page 119). Default false, will use current date.
	UseModificationDateTime (see page 121)	Switch to activate the ModificationDateTime (see page 120) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by ModificationDateTime (see page 120). Default false, will use current date.
	VolumeSet (see page 121)	Get/Set the Volume Set Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.1 ISOExOptions::AddSuffix Property

C++

```
property bool AddSuffix;
```

C#

```
public bool AddSuffix;
```

Description

Get/Set the bool value to enable ISO version number extension (;1).

The ISO 9660 specification requires that each filename include a ;x version number as the suffix. A full filename in ISO 9660 has three parts:

<name>.<extension>;<version>

Presumably the version part was designed to allow multiple versions of the same file to coexist.

1.1.1.21.2.2 ISOExOptions::AllowLongISO9660Names Property

C++

```
property bool AllowLongISO9660Names;
```

C#

```
public bool AllowLongISO9660Names;
```

Description

If enabled (true), allows long file and directory names up to 207 ASCII characters in ISO9660 file system. If options is disabled (false), the ISO Level restrictions on file name length will be used.

1.1.1.21.2.3 ISOExOptions::AllowLongJolietNames Property

C++

```
property bool AllowLongJolietNames;
```

C#

```
public bool AllowLongJolietNames;
```

Description

If enabled (true), allows long file and directory names up to 103 UCS-2 characters in Joliet extension to ISO9660 file system. If options is disabled (false), the max file name length will be limited by 64 UCS-2 characters.

1.1.1.21.2.4 ISOExOptions::AllowLowercaseNames Property

C++

```
property bool AllowLowercaseNames;
```

C#

```
public bool AllowLowercaseNames;
```

Description

Get/Set the bool value to allow lower case letters in file names.

1.1.1.21.2.5 ISOExOptions::AllowManyDirectories Property

C++

```
property bool AllowManyDirectories;
```

C#

```
public bool AllowManyDirectories;
```

Description

Get/Set the bool value to allow path dept of more than 8 directories.

1.1.1.21.2.6 ISOExOptions::ApplicationIdentifier Property

C++

```
property String ^ ApplicationIdentifier;
```

C#

```
public String ApplicationIdentifier;
```

Description

Get/Set the Volume Application Identifier field of the ISO/Joliet image(128 characters maximum).For detailed information see: ISO 9660

1.1.1.21.2.7 ISOExOptions::BiblioIdentifier Property

C++

```
property String ^ BiblioIdentifier;
```

C#

```
public String BiblioIdentifier;
```

Description

Get/Set the Volume Bibliography File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.8 ISOExOptions::CopyrightFile Property

C++

```
property String ^ CopyrightFile;
```

C#

```
public String CopyrightFile;
```

Description

Get/Set the Volume Copyright File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.9 ISOExOptions::CreationDateTime Property

C++

```
property FileDateTime ^ CreationDateTime;
```

C#

```
public FileDateTime CreationDateTime;
```

Description

The creation date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.

Notes

This value will only be used if UseCreationDateTime (see page 121) is set to true.

1.1.1.21.2.10 ISOExOptions::DataPreparer Property

C++

```
property String ^ DataPreparer;
```

C#

```
public String DataPreparer;
```

Description

Get/Set the Volume Data Preparer Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.11 ISOExOptions::EffectiveDateTime Property

C++

```
property FileDateTime ^ EffectiveDateTime;
```

C#

```
public FileDateTime EffectiveDateTime;
```

Description

The effective date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.

Notes

This value will only be used if UseEffectiveDateTime (see page 121) is set to true.

1.1.1.21.2.12 ISOExOptions::ExpirationDateTime Property

C++

```
property FileDateTime ^ ExpirationDateTime;
```

C#

```
public FileDateTime ExpirationDateTime;
```

Description

The expiration date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.

Notes

This value will only be used if UseExpirationDateTime (see page 121) is set to true.

1.1.1.21.2.13 ISOExOptions::FileIdentifier Property

C++

```
property String ^ FileIdentifier;
```

C#

```
public String FileIdentifier;
```

Description

Get/Set the Volume Abstract File Identifier field of the ISO/Joliet image (35 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.14 ISOExOptions::ISOLevel Property

C++

```
property IsoSDK::ISOLevel ISOLevel;
```

C#

```
public IsoSDK::ISOLevel ISOLevel;
```

Description

Get/set the ISOLevel for ISO9660 according to the ISOLevel enumeration.

1.1.1.21.2.15 ISOExOptions::ModificationDateTime Property

C++

```
property FileDateTime ^ ModificationDateTime;
```

C#

```
public FileDateTime ModificationDateTime;
```

Description

The modification date of the media according the ISO specification. Use a FileDateTime (see page 101) structure as value.

Notes

This value will only be used if UseModificationDateTime (see page 121) is set to true.

1.1.1.21.2.16 ISOExOptions::Publisher Property

C++

```
property String ^ Publisher;
```

C#

```
public String Publisher;
```

Description

Get/Set the Volume Publisher Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.17 ISOExOptions::SystemIdentifier Property

C++

```
property String ^ SystemIdentifier;
```

C#

```
public String SystemIdentifier;
```

Description

Get/Set the Volume System Identifier field of the ISO/Joliet image (31 characters maximum). For detailed information see: ISO 9660

1.1.1.21.2.18 ISOExOptions::UseCreationDateTime Property

C++

```
property bool UseCreationDateTime;
```

C#

```
public bool UseCreationDateTime;
```

Description

Switch to activate the CreationDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by CreationDateTime (see page 119). Default false, will use current date.

1.1.1.21.2.19 ISOExOptions::UseEffectiveDateTime Property

C++

```
property bool UseEffectiveDateTime;
```

C#

```
public bool UseEffectiveDateTime;
```

Description

Switch to activate the EffectiveDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by EffectiveDateTime (see page 119). Default false, will use current date.

1.1.1.21.2.20 ISOExOptions::UseExpirationDateTime Property

C++

```
property bool UseExpirationDateTime;
```

C#

```
public bool UseExpirationDateTime;
```

Description

Switch to activate the ExpirationDateTime (see page 119) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by ExpirationDateTime (see page 119). Default false, will use current date.

1.1.1.21.2.21 ISOExOptions::UseModificationDateTime Property

C++

```
property bool UseModificationDateTime;
```

C#

```
public bool UseModificationDateTime;
```

Description

Switch to activate the ModificationDateTime (see page 120) in the ISO structure. If set to true the IsoSDK (see page 1) will use the value setted by ModificationDateTime (see page 120). Default false, will use current date.

1.1.1.21.2.22 ISOExOptions::VolumeSet Property

C++

```
property String ^ VolumeSet;
```

C#

```
public String VolumeSet;
```

Description

Get/Set the Volume Set Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660

1.1.1.22 NetworkTags Class

Class Hierarchy

IsoSDK::NetworkTags

C++

```
ref class NetworkTags;
```

C#

```
public class NetworkTags;
```

File

IsoSDKBurnerNet.h

Description

This class will used to manage NetworkTags information. The NetWorkTags are the information of the Disk/Track received from a CDDB/FreeDB database.

1.1.1.22.1 NetworkTags Members

The following tables list the members exposed by NetworkTags.

Public Methods

	Name	Description
	GetNetworkDiskTagInt (see page 123)	This function will received a numeric value of a CDDB Disk-Tag.
	GetNetworkDiskTagString (see page 123)	This function will received a string value of a CDDB Disk-Tag.
	GetNetworkTrackTagInt (see page 123)	This function will received a numeric value of a CDDB Track-Tag according to the given track.
	GetNetworkTrackTagString (see page 124)	This function will received a string value of a CDDB Track-Tag according to the given track.

1.1.1.22.2 NetworkTags Methods

The methods of the NetworkTags class are listed here.

Public Methods

	Name	Description
	GetNetworkDiskTagInt (see page 123)	This function will received a numeric value of a CDDB Disk-Tag.
	GetNetworkDiskTagString (see page 123)	This function will received a string value of a CDDB Disk-Tag.
	GetNetworkTrackTagInt (see page 123)	This function will received a numeric value of a CDDB Track-Tag according to the given track.
	GetNetworkTrackTagString (see page 124)	This function will received a string value of a CDDB Track-Tag according to the given track.

1.1.1.22.2.1 NetworkTags::GetNetworkDiskTagInt Method

C++

```
int GetNetworkDiskTagInt(
    NetworkTagsContentItem NTCI
);
```

C#

```
public int GetNetworkDiskTagInt(
    NetworkTagsContentItem NTCI
);
```

Parameters

Parameters	Description
NetworkTagsContentItem NTCI	The CDDB item to received according to the NetworkTagsContentItem (see page 194) enumeration.

Description

This function will received a numeric value of a CDDB Disk-Tag.

1.1.1.22.2.2 NetworkTags::GetNetworkDiskTagString Method

C++

```
String ^ GetNetworkDiskTagString(
    NetworkTagsContentItem NTCI
);
```

C#

```
public String GetNetworkDiskTagString(
    NetworkTagsContentItem NTCI
);
```

Parameters

Parameters	Description
NetworkTagsContentItem NTCI	The CDDB item to received according to the NetworkTagsContentItem (see page 194) enumeration.

Description

This function will received a string value of a CDDB Disk-Tag.

1.1.1.22.2.3 NetworkTags::GetNetworkTrackTagInt Method

C++

```
int GetNetworkTrackTagInt(
    int nTrackNumber,
    NetworkTagsContentItem NTCI
);
```

C#

```
public int GetNetworkTrackTagInt(
    int nTrackNumber,
    NetworkTagsContentItem NTCI
);
```

Parameters

Parameters	Description
int nTrackNumber	Number of the track to receive data from
NetworkTagsContentItem NTCI	The CDDB item to received according to the NetworkTagsContentItem (see page 194) enumeration.

Description

This function will received a numeric value of a CDDB Track-Tag according to the given track.

1.1.1.22.2.4 NetworkTags::GetNetworkTrackTagString Method

C++

```
String ^ GetNetworkTrackTagString(
    int nTrackNumber,
    NetworkTagsContentItem NTCI
);
```

C#

```
public String GetNetworkTrackTagString(
    int nTrackNumber,
    NetworkTagsContentItem NTCI
);
```

Parameters

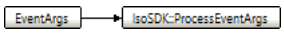
Parameters	Description
int nTrackNumber	Number of the track to receive data from.
NetworkTagsContentItem NTCI	The CDDB item to received according to the NetworkTagsContentItem (see page 194) enumeration.

Description

This function will received a string value of a CDDB Track-Tag according to the given track.

1.1.1.23 ProcessEventArgs Class

Class Hierarchy



C++

```
ref class ProcessEventArgs : public EventArgs;
```

C#

```
public class ProcessEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the Process event.

1.1.1.23.1 ProcessEventArgs Members

The following tables list the members exposed by ProcessEventArgs.

Public Data Members

	Name	Description
	m_dBytesWritten (see page 125)	A double value with the current written bytes.
	m_dImageSize (see page 125)	A double value with the total disk size.
	m_fCache (see page 125)	Cache usage in % (0 – 100).
	m_fDeviceBuffer (see page 125)	Device buffer usage in % (0 – 100).
	m_fPercent (see page 126)	A float value with the total progress in % (0 – 100).

1.1.1.23.2 ProcessEventArgs Data Members

The data members of the ProcessEventArgs class are listed here.

Public Data Members

	Name	Description
	m_dBytesWritten (see page 125)	A double value with the current written bytes.
	m_dImageSize (see page 125)	A double value with the total disk size.
	m_fCache (see page 125)	Cache usage in % (0 – 100).
	m_fDeviceBuffer (see page 125)	Device buffer usage in % (0 – 100).
	m_fPercent (see page 126)	A float value with the total progress in % (0 – 100).

1.1.1.23.2.1 ProcessEventArgs::m_dBytesWritten Data Member

C++

```
double m_dBytesWritten;
```

C#

```
public double m_dBytesWritten;
```

Description

A double value with the current written bytes.

1.1.1.23.2.2 ProcessEventArgs::m_dImageSize Data Member

C++

```
double m_dImageSize;
```

C#

```
public double m_dImageSize;
```

Description

A double value with the total disk size.

1.1.1.23.2.3 ProcessEventArgs::m_fCache Data Member

C++

```
float m_fCache;
```

C#

```
public float m_fCache;
```

Description

Cache usage in % (0 – 100).

1.1.1.23.2.4 ProcessEventArgs::m_fDeviceBuffer Data Member

C++

```
float m_fDeviceBuffer;
```

C#

```
public float m_fDeviceBuffer;
```

Description

Device buffer usage in % (0 – 100).

1.1.1.23.2.5 ProcessEventArgs::m_fPercent Data Member

C++

```
float m_fPercent;
```

C#

```
public float m_fPercent;
```

Description

A float value with the total progress in % (0 – 100).

1.1.1.24 RemoveFileEventArgs Class

Class Hierarchy

C++

```
ref class RemoveFileEventArgs : public EventArgs;
```

C#

```
public class RemoveFileEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the RemoveFile event.

1.1.1.24.1 RemoveFileEventArgs Members

The following tables list the members exposed by RemoveFileEventArgs.

Public Data Members

	Name	Description
	m_strDestinationPath (see page 126)	A string with the destination path of the file in the project.
	m_strFileName (see page 127)	A string with the source path of the removed file (includes file name).
	m_strFullPath (see page 127)	A string with the path of the removed file (includes file name).

1.1.1.24.2 RemoveFileEventArgs Data Members

The data members of the RemoveFileEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strDestinationPath (see page 126)	A string with the destination path of the file in the project.
	m_strFileName (see page 127)	A string with the source path of the removed file (includes file name).
	m_strFullPath (see page 127)	A string with the path of the removed file (includes file name).

1.1.1.24.2.1 RemoveFileEventArgs::m_strDestinationPath Data Member

C++

```
String ^ m_strDestinationPath;
```

C#

```
public String m_strDestinationPath;
```

Description

A string with the destination path of the file in the project.

1.1.1.24.2.2 RemoveFileEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

```
public String m_strFileName;
```

Description

A string with the source path of the removed file (includes file name).

1.1.1.24.2.3 RemoveFileEventArgs::m_strFullPath Data Member

C++

```
String ^ m_strFullPath;
```

C#

```
public String m_strFullPath;
```

Description

A string with the path of the removed file (includes file name).

1.1.1.25 TextEventArgs Class

Class Hierarchy

C++

```
ref class TextEventArgs : public EventArgs;
```

C#

```
public class TextEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the Text event.

1.1.1.25.1 TextEventArgs Members

The following tables list the members exposed by TextEventArgs.

Public Data Members

	Name	Description
	m_nTextID (see page 128)	This value is the Text-ID of the required text.
	m_pnLength (see page 128)	This is the text length of m_strText (see page 128).
	m_strText (see page 128)	A string with the corresponding text of the reported ID.

1.1.1.25.2 TextEventArgs Data Members

The data members of the TextEventArgs class are listed here.

Public Data Members

	Name	Description
	m_nTextID (see page 128)	This value is the Text-ID of the required text.
	m_pnLength (see page 128)	This is the text length of m_strText (see page 128).
	m_strText (see page 128)	A string with the corresponding text of the reported ID.

1.1.1.25.2.1 TextEventArgs::m_nTextID Data Member

C++

```
int m_nTextID;
```

C#

```
public int m_nTextID;
```

Description

This value is the Text-ID of the required text.

1.1.1.25.2.2 TextEventArgs::m_pnLength Data Member

C++

```
int * m_pnLength;
```

C#

```
public int * m_pnLength;
```

Description

This is the text length of m_strText (see page 128).

1.1.1.25.2.3 TextEventArgs::m_strText Data Member

C++

```
String ^ m_strText;
```

C#

```
public String m_strText;
```

Description

A string with the corresponding text of the reported ID.

1.1.1.26 UDFOptions Class

Class Hierarchy

IsoSDK::UDFOptions

C++

```
ref class UDFOptions;
```

C#

```
public class UDFOptions;
```

File

Options.h

Description

This class will used to manage UDFOptions information. A instance of this class will created on call of the Property UDFOptions.

1.1.1.26.1 UDFOptions Members

The following tables list the members exposed by UDFOptions.

Public Properties

	Name	Description
	ImplementationID (see page 129)	This is the volume name of a UDF file system disk. Length max. 23 characters.
	IsAvchdDisc (see page 129)	Allow to burn the Blu-ray file system to a DVD blank.
	PartitionType (see page 130)	UDF partition type according to the UDF Partition Type enumeration. Only available with UDF version > 1.02
	Version (see page 130)	UDF version according to the UDF Version enumeration.
	WriteStreams (see page 130)	States whether file streams should be used. Only available with UDF >= 2.0

1.1.1.26.2 UDFOptions Properties

The properties of the UDFOptions class are listed here.

Public Properties

	Name	Description
	ImplementationID (see page 129)	This is the volume name of a UDF file system disk. Length max. 23 characters.
	IsAvchdDisc (see page 129)	Allow to burn the Blu-ray file system to a DVD blank.
	PartitionType (see page 130)	UDF partition type according to the UDF Partition Type enumeration. Only available with UDF version > 1.02
	Version (see page 130)	UDF version according to the UDF Version enumeration.
	WriteStreams (see page 130)	States whether file streams should be used. Only available with UDF >= 2.0

1.1.1.26.2.1 UDFOptions::ImplementationID Property

C++

```
property String ^ ImplementationID;
```

C#

```
public String ImplementationID;
```

Description

This is the volume name of a UDF file system disk. Length max. 23 characters.

1.1.1.26.2.2 UDFOptions::IsAvchdDisc Property

C++

```
property bool IsAvchdDisc;
```

C#

```
public bool IsAvchdDisc;
```

Description

Allow to burn the Blu-ray file system to a DVD blank.

1.1.1.26.2.3 UDFOptions::PartitionType Property

C++

```
property UDFPartitionType PartitionType;
```

C#

```
public UDFPartitionType PartitionType;
```

Description

UDF partition type according to the UDF Partition Type enumeration.

Only available with UDF version > 1.02

1.1.1.26.2.4 UDFOptions::Version Property

C++

```
property UDFVersion Version;
```

C#

```
public UDFVersion Version;
```

Description

UDF version according to the UDF Version enumeration.

1.1.1.26.2.5 UDFOptions::WriteStreams Property

C++

```
property bool WriteStreams;
```

C#

```
public bool WriteStreams;
```

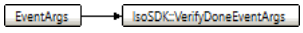
Description

States whether file streams should be used.

Only available with UDF >= 2.0

1.1.1.27 VerifyDoneEventArgs Class

Class Hierarchy



C++

```
ref class VerifyDoneEventArgs : public EventArgs;
```

C#

```
public class VerifyDoneEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the VerifyDone event.

1.1.1.27.1 VerifyDoneEventArgs Members

The following tables list the members exposed by VerifyDoneEventArgs.

Public Data Members

	Name	Description
	m_nNumErrors (see page 131)	Counted errors during the verify process.

1.1.1.27.2 VerifyDoneEventArgs Data Members

The data members of the VerifyDoneEventArgs class are listed here.

Public Data Members

	Name	Description
	m_nNumErrors (see page 131)	Counted errors during the verify process.

1.1.1.27.2.1 VerifyDoneEventArgs::m_nNumErrors Data Member

C++

```
int m_nNumErrors;
```

C#

```
public int m_nNumErrors;
```

Description

Counted errors during the verify process.

1.1.1.28 VerifyErrorEventArgs Class

Class Hierarchy

C++

```
ref class VerifyErrorEventArgs : public EventArgs;
```

C#

```
public class VerifyErrorEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the VerifyError event.

1.1.1.28.1 VerifyErrorEventArgs Members

The following tables list the members exposed by VerifyErrorEventArgs.

Public Data Members

	Name	Description
	m_strError (see page 132)	A string with the description of the error.
	m_strFileName (see page 132)	A string with the file name and path of the current file in process.

1.1.1.28.2 VerifyErrorEventArgs Data Members

The data members of the VerifyErrorEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strError (see page 132)	A string with the description of the error.
	m_strFileName (see page 132)	A string with the file name and path of the current file in process.

1.1.1.28.2.1 VerifyErrorEventArgs::m_strError Data Member

C++

```
String ^ m_strError;
```

C#

```
public String m_strError;
```

Description

A string with the description of the error.

1.1.1.28.2.2 VerifyErrorEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

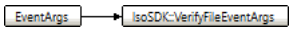
```
public String m_strFileName;
```

Description

A string with the file name and path of the current file in process.

1.1.1.29 VerifyFileEventArgs Class

Class Hierarchy



C++

```
ref class VerifyFileEventArgs : public EventArgs;
```

C#

```
public class VerifyFileEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the VerifyFile event.

1.1.1.29.1 VerifyFileEventArgs Members

The following tables list the members exposed by VerifyFileEventArgs.

Public Data Members

	Name	Description
	m_strFileName (see page 133)	A string with the file name and path of the current file in process.

1.1.1.29.2 VerifyFileEventArgs Data Members

The data members of the VerifyFileEventArgs class are listed here.

Public Data Members

	Name	Description
	m_strFileName (see page 133)	A string with the file name and path of the current file in process.

1.1.1.29.2.1 VerifyFileEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

```
public String m_strFileName;
```

Description

A string with the file name and path of the current file in process.

1.1.1.30 VerifySectorEventArgs Class

Class Hierarchy

C++

```
ref class VerifySectorEventArgs : public EventArgs;
```

C#

```
public class VerifySectorEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the VerifySector event.

1.1.1.30.1 VerifySectorEventArgs Members

The following tables list the members exposed by VerifySectorEventArgs.

Public Data Members

	Name	Description
	nSector (see page 134)	Number of the current sector.
	nSuccess (see page 134)	Current sector valid or invalid.
	tSector (see page 134)	Number of all sectors.

1.1.1.30.2 VerifySectorEventArgs Data Members

The data members of the VerifySectorEventArgs class are listed here.

Public Data Members

	Name	Description
	nSector (see page 134)	Number of the current sector.
	nSuccess (see page 134)	Current sector valid or invalid.
	tSector (see page 134)	Number of all sectors.

1.1.1.30.2.1 VerifySectorEventArgs::nSector Data Member

C++

```
double nSector;
```

C#

```
public double nSector;
```

Description

Number of the current sector.

1.1.1.30.2.2 VerifySectorEventArgs::nSuccess Data Member

C++

```
bool nSuccess;
```

C#

```
public bool nSuccess;
```

Description

Current sector valid or invalid.

1.1.1.30.2.3 VerifySectorEventArgs::tSector Data Member

C++

```
double tSector;
```

C#

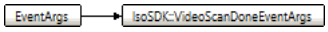
```
public double tSector;
```

Description

Number of all sectors.

1.1.1.31 VideoScanDoneEventArgs Class

Class Hierarchy



C++

```
ref class VideoScanDoneEventArgs : public EventArgs;
```

C#

```
public class VideoScanDoneEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the VideoScanDone event.

1.1.1.31.1 VideoScanDoneEventArgs Members

The following tables list the members exposed by VideoScanDoneEventArgs.

Public Data Members

	Name	Description
	m_nAspectRatio (see page 135)	The type of the aspect ratio according to the AspectRatio (see page 140) enumeration.
	m_nBitRate (see page 136)	The current bitrate of the video in bits / seconds (bits/s).
	m_nErrorCode (see page 136)	Error ID when an error occurred. See Language file.
	m_nFPS (see page 136)	A double with the current frame rate in frames / second (FPS).
	m_nHeight (see page 136)	The height of the video in pixel.
	m_nPlayTime (see page 136)	The playtime of the video in seconds.
	m_nWidth (see page 137)	The width of the video in pixel.
	m_strError (see page 137)	A string with the decoding error. If no error occurred, an empty string or a null pointer will be returned.
	m_strFileName (see page 137)	A string with the file name and path of the video file that was scanned.

1.1.1.31.2 VideoScanDoneEventArgs Data Members

The data members of the VideoScanDoneEventArgs class are listed here.

Public Data Members

	Name	Description
	m_nAspectRatio (see page 135)	The type of the aspect ratio according to the AspectRatio (see page 140) enumeration.
	m_nBitRate (see page 136)	The current bitrate of the video in bits / seconds (bits/s).
	m_nErrorCode (see page 136)	Error ID when an error occurred. See Language file.
	m_nFPS (see page 136)	A double with the current frame rate in frames / second (FPS).
	m_nHeight (see page 136)	The height of the video in pixel.
	m_nPlayTime (see page 136)	The playtime of the video in seconds.
	m_nWidth (see page 137)	The width of the video in pixel.
	m_strError (see page 137)	A string with the decoding error. If no error occurred, an empty string or a null pointer will be returned.
	m_strFileName (see page 137)	A string with the file name and path of the video file that was scanned.

1.1.1.31.2.1 VideoScanDoneEventArgs::m_nAspectRatio Data Member

C++

```
AspectRatio m_nAspectRatio;
```

C#

```
public AspectRatio m_nAspectRatio;
```

Description

The type of the aspect ratio according to the AspectRatio (see page 140) enumeration.

1.1.1.31.2.2 VideoScanDoneEventArgs::m_nBitRate Data Member

C++

```
int m_nBitRate;
```

C#

```
public int m_nBitRate;
```

Description

The current bitrate of the video in bits / seconds (bits/s).

1.1.1.31.2.3 VideoScanDoneEventArgs::m_nErrorCode Data Member

C++

```
ErrorCode m_nErrorCode;
```

C#

```
public ErrorCode m_nErrorCode;
```

Description

Error ID when an error occurred. See Language file.

1.1.1.31.2.4 VideoScanDoneEventArgs::m_nFPS Data Member

C++

```
double m_nFPS;
```

C#

```
public double m_nFPS;
```

Description

A double with the current frame rate in frames / second (FPS).

1.1.1.31.2.5 VideoScanDoneEventArgs::m_nHeight Data Member

C++

```
int m_nHeight;
```

C#

```
public int m_nHeight;
```

Description

The height of the video in pixel.

1.1.1.31.2.6 VideoScanDoneEventArgs::m_nPlayTime Data Member

C++

```
int m_nPlayTime;
```

C#

```
public int m_nPlayTime;
```

Description

The playtime of the video in seconds.

1.1.1.31.2.7 VideoScanDoneEventArgs::m_nWidth Data Member

C++

```
int m_nWidth;
```

C#

```
public int m_nWidth;
```

Description

The width of the video in pixel.

1.1.1.31.2.8 VideoScanDoneEventArgs::m_strError Data Member

C++

```
String ^ m_strError;
```

C#

```
public String m_strError;
```

Description

A string with the decoding error. If no error occurred, an empty string or a null pointer will be returned.

1.1.1.31.2.9 VideoScanDoneEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

```
public String m_strFileName;
```

Description

A string with the file name and path of the video file that was scanned.

1.1.1.32 VideoScannerEventArgs Class

Class Hierarchy

C++

```
ref class VideoScannerEventArgs : public EventArgs;
```

C#

```
public class VideoScannerEventArgs : EventArgs;
```

File

EventArgs.h

Description

Provides data and arguments for the VideoScanner event.

1.1.1.32.1 VideoScannerEventArgs Members

The following tables list the members exposed by VideoScannerEventArgs.

Public Data Members

	Name	Description
	m_fPercent (see page 138)	A float with the scanning progress in % (0 – 100).
	m_strFileName (see page 138)	A string with the file name and path of the video file that is being scanned.

1.1.1.32.2 VideoScannerEventArgs Data Members

The data members of the VideoScannerEventArgs class are listed here.

Public Data Members

	Name	Description
	m_fPercent (see page 138)	A float with the scanning progress in % (0 – 100).
	m_strFileName (see page 138)	A string with the file name and path of the video file that is being scanned.

1.1.1.32.2.1 VideoScannerEventArgs::m_fPercent Data Member

C++

```
float m_fPercent;
```

C#

```
public float m_fPercent;
```

Description

A float with the scanning progress in % (0 – 100).

1.1.1.32.2.2 VideoScannerEventArgs::m_strFileName Data Member

C++

```
String ^ m_strFileName;
```

C#

```
public String m_strFileName;
```

Description

A string with the file name and path of the video file that is being scanned.

1.1.2 Structs, Records, Enums

The following table lists structs, records, enums in this documentation.

Enumerations

	Name	Description
	AspectRatio (see page 140)	This enumeration defines the possible Aspect Ratios of a MPEG Movie the IsoSDK (see page 1) supports.
	ASPIInterface (see page 141)	This enumeration defines the ASPI interfaces the IsoSDK (see page 1) supports.
	AudioFormat (see page 142)	This enumeration defines the audio formats the IsoSDK (see page 1) supports for encoding.
	BitrateType (see page 144)	This enumeration defines the bitrate types the IsoSDK (see page 1) supports.
	Capabilities (see page 146)	This enumeration defines the capabilities of the device. Please click on the feature you want to get information for.

	CDTextContentItem (see page 151)	This enumeration defines the types of CD-Text information the IsoSDK (see page 1) supports.
	DeviceIndex (see page 153)	This enumeration defines the possible selection for devices.
new	DiscSignature (see page 154)	This enumeration defines the compression and/or encryption state of the disc or image.
	ErrorCode (see page 155)	This enumeration defines the error codes the IsoSDK (see page 1) can throw out.
	ExtendedMediumType (see page 182)	This enumeration defines the extended type of the current medium.
	FileAttributes (see page 184)	This enumeration defines the file attribute of a file to add.
	FileSystems (see page 186)	This enumeration defines the file system of a medium the IsoSDK (see page 1) supports.
	ImageFormat (see page 186)	This enumeration defines the image formats the IsoSDK (see page 1) supports for writing.
	ImageTask (see page 187)	This enumeration defines the possible actions for image creation task while disk copy.
	InfoLevel (see page 187)	This enumeration defines the level of informations that the IsoSDK (see page 1) supports.
	ISOLevel (see page 188)	This enumeration defines the possible extended ISO9660 derivate the IsoSDK (see page 1) supports.
	MediumStatus (see page 191)	This enumeration defines the status of the current medium.
	MediumType (see page 192)	This enumeration defines the type of the current medium.
	NetworkTagsContentItem (see page 194)	This enumeration defines the CDDB fields that are supported by the IsoSDK (see page 1).
	ProjectType (see page 195)	This enumeration defines the project types the IsoSDK (see page 1) supports.
	RawDataType (see page 196)	This enumeration defines the data type for a RAW Project type.
	RawTrackFormat (see page 197)	This enumeration defines the track format type for a RAW project type.
	ReadMode (see page 198)	This enumeration defines the read mode the IsoSDK (see page 1) supports.
	SavePathOption (see page 199)	This enumeration defines if and how the super ordinate directory is added as a folder to the project.
	SaveTrackFileFormat (see page 200)	This enumeration defines the possible formats the IsoSDK (see page 1) supports to save tracks.
	SessionStatus (see page 201)	This enumeration defines the status of the last session.
	TagChoiceType (see page 202)	This enumeration defines the type of tags the IsoSDK (see page 1) will try to receive.
	TrackFormat (see page 202)	This enumeration defines the track type of a medium.
	UDFPartitionType (see page 204)	This enumeration defines the UDF partition the IsoSDK (see page 1) supports.
	UDFVersion (see page 205)	This enumeration defines the UDF version the IsoSDK (see page 1) supports.
	WriteMethod (see page 207)	This enumeration defines the write methods the IsoSDK (see page 1) supports.

Structures

	Name	Description
	AudioFileProperty (see page 141)	This structure contains information about CD-Text.
	AudioGrabbingParams (see page 143)	A structure that contains information for Audiograbbing
	BootVolumeInfo (see page 144)	A structure that contains the information about the boot volume information.
	BurnIsoOptions (see page 145)	A structure that contains the information that are needed to burn an ISO file.

	CreateImageParams (see page 152)	A structure that contains information for the ImageCreate operation.
	DeviceInformation (see page 153)	A structure that contains device information of a device from the internal device list.
	DiskCopyOptions (see page 154)	A structure that contains information for the disk copy operation.
	ExtendedDeviceInformation (see page 181)	A structure that contains the extended information for the device.
	Extent (see page 183)	A structure that contains the information about the Extent information of a file in allocation table.
	FileAllocationTable (see page 183)	A structure that contains the information about the file allocation table.
	FileEntry (see page 185)	A structure that contains the information for the file to be used.
	ISOVolumeInfo (see page 189)	A structure that contains the information about the ISO information.
	MediumInfo (see page 190)	A structure that contains the information about the medium.
	RawTrack (see page 196)	This structure contains the information of a RAW image.
	ReadErrorCorrectionParams (see page 198)	A structure that contains the information of the ReadErrorCorrection for CopyDisk and CreateImage.
	SessionInfo (see page 200)	A structure that contains the information about the selected session.
	Speed (see page 201)	This structure contains information about the possible burning speeds.
	TrackInfo (see page 203)	A structure that contains the information about the selected track.
	UDFVolumeInfo (see page 206)	A structure that contains the information about the UDF information.

1.1.2.1 IsoSDK::AspectRatio Enumeration

C++

```
enum class AspectRatio {
  SquarePixels = 0,
  Display4To3 = 1,
  Display16To9 = 2,
  Display221To2 = 3,
  Unknown = 4
};
```

C#

```
public enum AspectRatio {
  SquarePixels = 0,
  Display4To3 = 1,
  Display16To9 = 2,
  Display221To2 = 3,
  Unknown = 4
}
```

File

EventArgs.h

Members

Members	Description
SquarePixels = 0	Mpeg file has square pixels.
Display4To3 = 1	The Mpeg file has a 4:3 display.
Display16To9 = 2	Mpeg file has a 16:9 display.
Display221To2 = 3	Mpeg file has a 2.21:1 display.
Unknown = 4	Mpeg file has an unknown aspect ratio.

Description

This enumeration defines the possible Aspect Ratios of a MPEG Movie the IsoSDK (see page 1) supports.

1.1.2.2 IsoSDK::ASPIInterface Enumeration

C++

```
enum class ASPIInterface {
  Internal = 0,
  WnAspi = 1,
  FrogAspi = 2
};
```

C#

```
public enum ASPIInterface {
  Internal = 0,
  WnAspi = 1,
  FrogAspi = 2
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Internal = 0	The IsoSDK (see page 1) internal ASPI. This is recommended for all Windows versions from Windows 2000 and above.
WnAspi = 1	The default external ASPI layer like the one from Adaptec. Please note that this ASPI does neither support USB nor FireWire.
FrogAspi = 2	The FrogAspi DLL. This is a very good replacement of the Adaptec interface. It supports all new interfaces.

Description

This enumeration defines the ASPI interfaces the IsoSDK (see page 1) supports.

1.1.2.3 IsoSDK::AudioFileProperty Structure

C++

```
ref struct AudioFileProperty {
  String ^ SourceFilePath;
  String ^ Title;
  String ^ Performer;
  String ^ SongWriter;
  String ^ Composer;
  String ^ Arranger;
  String ^ Message;
  String ^ McnIsrc;
  int Pause;
  bool PauseInFrames;
  array<int> ^ Indexes;
  int IndexesLength;
};
```

C#

```
public struct AudioFileProperty {
  public String SourceFilePath;
  public String Title;
  public String Performer;
  public String SongWriter;
  public String Composer;
  public String Arranger;
  public String Message;
  public String McnIsrc;
```

```
    public int Pause;
    public bool PauseInFrames;
    public array<int> Indexes;
    public int IndexesLength;
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ SourceFilePath;	Pointer to the null-terminated string with the source path of the audio file which properties you want to change; or the root directory of audio files ("\" for audio projects and "\audio" for Mixed Mode projects) if you want to change the properties for a disk.
String ^ Title;	The title of the track/disk (CD-TEXT).
String ^ Performer;	The performer of the track/disk (CD-TEXT).
String ^ SongWriter;	The song writer of the track/disk (CD-TEXT).
String ^ Composer;	The composer of the track/disk (CD-TEXT).
String ^ Arranger;	The arranger of the track/disk (CD-TEXT).
String ^ Message;	The message of the track/disk (CD-TEXT).
String ^ McnIsrc;	MCN or ISRC depending on lpszSourceFilePath TRACK = ISRC = International Standard Recording Code (http://www.ifpi.org/) DISK = MCN = Media Catalog Number
int Pause;	The pause size before a track in seconds. Ignored for disk
bool PauseInFrames;	Switch between frames / seconds value for Pause property.
array<int> ^ Indexes;	A pointer to a n array of indexes inside track. This are the sub indexes for the Track only, not for the disk.
int IndexesLength;	Number of items in pIndexes.

Description

This structure contains information about CD-Text.

1.1.2.4 IsoSDK::AudioFormat Enumeration

C++

```
  enum class AudioFormat {
    Mp3 = 0,
    Aac = 1,
    Ogg = 2,
    Opus = 3,
    Flac = 4
  };
```

C#

```
  public enum AudioFormat {
    Mp3 = 0,
    Aac = 1,
    Ogg = 2,
    Opus = 3,
    Flac = 4
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Mp3 = 0	Encoding type: MP3 (MPEG Audio Layer 3).
Aac = 1	Encoding type: AAC (Advanced Audio Codec)
Ogg = 2	Encoding type: Ogg Vorbis
Opus = 3	Encoding type: Opus
Flac = 4	Encoding type: FLAC (Free Lossless Audio Codec).

Description

This enumeration defines the audio formats the IsoSDK (see page 1) supports for encoding.

1.1.2.5 IsoSDK::AudioGrabbingParams Structure

C++

```
ref struct AudioGrabbingParams {
  unsigned int Bitrate;
  unsigned int MinBitrate;
  unsigned int MaxBitrate;
  unsigned int Quality;
  BitrateType BitrateType;
  AudioFormat EncoderType;
  TagChoiceType TagChoice;
  NetworkTags ^ NetworkTags;
};
```

C#

```
public struct AudioGrabbingParams {
  public unsigned int Bitrate;
  public unsigned int MinBitrate;
  public unsigned int MaxBitrate;
  public unsigned int Quality;
  public BitrateType BitrateType;
  public AudioFormat EncoderType;
  public TagChoiceType TagChoice;
  public NetworkTags NetworkTags;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
unsigned int Bitrate;	The bitrate value for the CBR encoding.
unsigned int MinBitrate;	The min, value for the VBR encoding.
unsigned int MaxBitrate;	The max, value for the VBR encoding.
unsigned int Quality;	A numeric value to define the Quality for audio encoding. For most codec the value is 1-6
BitrateType BitrateType;	The type how the IsoSDK (see page 1) will encode the file according to the BitrateType enumeration.
AudioFormat EncoderType;	The final target Encoding format according to the AudioFormat (see page 142) enumeration.
TagChoiceType TagChoice;	The type how the IsoSDK (see page 1) will write the Tags according to the TagChoiceType (see page 202) enumeration.
NetworkTags ^ NetworkTags;	Information of internet data base tags container according to the NetworkTags class.

Description

A structure that contains information for Audiograbbing

1.1.2.6 IsoSDK::BitrateType Enumeration

C++

```
enum class BitrateType {
  Variable = 0,
  Constant = 1,
  Average = 2
};
```

C#

```
public enum BitrateType {
  Variable = 0,
  Constant = 1,
  Average = 2
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Variable = 0	Audio encoding: Variable Bitrate (VBR)
Constant = 1	Audio encoding: Constant Bitrate (CBR)
Average = 2	Audio encoding: Average Bitrate (CBR)

Description

This enumeration defines the bitrate types the IsoSDK (see page 1) supports.

1.1.2.7 IsoSDK::BootVolumeInfo Structure

C++

```
ref struct BootVolumeInfo {
  long VolumeDescriptorAddress;
  String ^ DeveloperID;
  bool BootIndicator;
  long LoadSegment;
  long PlatformID;
  long Emulation;
  long SectorCount;
};
```

C#

```
public struct BootVolumeInfo {
  public long VolumeDescriptorAddress;
  public String DeveloperID;
  public bool BootIndicator;
  public long LoadSegment;
  public long PlatformID;
  public long Emulation;
  public long SectorCount;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
long VolumeDescriptorAddress;	The address where this descriptor starts.
String ^ DeveloperID;	These are the bytes 4-1F of the header entry. It contains the string to the developer. E. g. your company name. Max. length 24 chars.
bool BootIndicator;	This is byte 0 of the boot catalog. True = bootable (0x88) and false = not bootable (0x00).
long LoadSegment;	These are the bytes 2-3 of the boot catalog. The default value is 1984 (7C0). It is the address of the load segment for the initial boot image.
long PlatformID;	The platform ID, byte 1 of the header entry: 0 = 80x86 1 = Power PC 2 = Mac
long Emulation;	This is byte 1 from the boot catalog. It sets the boot media type. It specifies what media the boot image is intended to emulate. 0 = No emulation 1 = 1.2 MB diskette 2 = 1.44 MB diskette 3 = 2.88 MB diskette 4 = hard disk
long SectorCount;	These are the bytes 6-7 of the boot catalog. They represent the number of virtual/emulated sectors the system will store at the load segment.

Description

A structure that contains the information about the boot volume information.

1.1.2.8 IsoSDK::BurnIsoOptions Structure

C++

```
ref struct BurnIsoOptions {
  WriteMethod WriteMethod;
  bool FinalizeDisk;
  bool TestBurn;
  bool PerformOPC;
  int CacheSize;
  bool UnderrunProtection;
  bool EjectAfterBurn;
  bool AutoErase;
};
```

C#

```
public struct BurnIsoOptions {
  public WriteMethod WriteMethod;
  public bool FinalizeDisk;
  public bool TestBurn;
  public bool PerformOPC;
  public int CacheSize;
  public bool UnderrunProtection;
  public bool EjectAfterBurn;
  public bool AutoErase;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
WriteMethod WriteMethod;	Get/Set the write method. The property will be filled according to the WriteMethod enumeration.
bool FinalizeDisk;	Get/Set whether the medium should be finalized. true finalizes the medium, whereas false burns a multi-session medium).
bool TestBurn;	Get/Set the simulation option. true activates the burning simulation and false skips the simulation.
bool PerformOPC;	Get/Set whether you want to perform OPC or not. Pass true to activate it and pass false to de-activate it. This option is not available for Video DVD projects.
int CacheSize;	Sets the size of the burning buffer to be used (in bytes; default: 4 * 1024 * 1024 [4 MB])
bool UnderrunProtection;	Get/Set whether the buffer-underrun-protection will be used.
bool EjectAfterBurn;	Get/Set whether the medium will be ejected after the burning process. true ejects the medium, and false prevents this.
bool AutoErase;	This argument sets the auto erase function for CD/DVD/BluRay ReWriteable. If this option is set to true the IsoSDK (see page 1) will erase a RW before the new burn job started. Please note that this option use the "Quick erase" function.

Description

A structure that contains the information that are needed to burn an ISO file.

1.1.2.9 IsoSDK::Capabilities Enumeration

C++

```
enum class Capabilities : System::Int64 {
  ReadCdR = 0x00000000001LL,
  ReadCdRw = 0x00000000002LL,
  ReadDvd = 0x00000000004LL,
  ReadDvdR = 0x00000000008LL,
  ReadDvdRw = 0x00000000010LL,
  ReadDvdRam = 0x00000000020LL,
  ReadDvdRPlus = 0x00000000040LL,
  ReadDvdRwPlus = 0x00000000080LL,
  ReadDvdDL = 0x00000000100LL,
  ReadDvdMrDL = 0x00000000200LL,
  ReadBlurayR = 0x00000000400LL,
  ReadBlurayRe = 0x00000000800LL,
  ReadBlurayRom = 0x00000001000LL,
  ReadHdDvdR = 0x00000002000LL,
  ReadHdDvdRw = 0x00000004000LL,
  ReadHdDvdRom = 0x00000008000LL,
  ReadMountRainer = 0x00000020000LL,
  ReadCdRwCav = 0x00000040000LL,
  WriteCdR = 0x00000080000LL,
  WriteCdRw = 0x00000100000LL,
  WriteDvdR = 0x00000200000LL,
  WriteDvdRw = 0x00000400000LL,
  WriteDvdRam = 0x00000800000LL,
  WriteDvdRPlus = 0x00001000000LL,
  WriteDvdRwPlus = 0x00002000000LL,
  WriteDvdDL = 0x00004000000LL,
  WriteDvdMrDL = 0x00008000000LL,
  WriteBlurayR = 0x00010000000LL,
  WriteBlurayRe = 0x00020000000LL,
  WriteHdDvdR = 0x00040000000LL,
  WriteHdDvdRw = 0x00080000000LL,
```

```
    WriteMountRainer = 0x00400000000LL,
    WriteCdRwCav = 0x00800000000LL,
    WriteTest = 0x01000000000LL,
    UnderrunProtection = 0x02000000000LL,
    Smart = 0x04000000000LL,
    Multisession = 0x08000000000LL,
    CprmAuth = 0x10000000000LL,
    DefectManagement = 0x20000000000LL,
    Streaming = 0x40000000000LL,
    ReadDvdRDLPlus = 0x080000000000LL,
    ReadDvdRwDLPlus = 0x100000000000LL,
    WriteDvdRDLPlus = 0x200000000000LL,
    WriteDvdRwDLPlus = 0x400000000000LL,
    LayerJumpRecording = 0x800000000000LL,
    AnalogAudioPlayback = 0xFFFF00000001LL,
    CompositeAudioAndVideo = 0xFFFF00000002LL,
    DigitalPort1 = 0xFFFF00000003LL,
    DigitalPort2 = 0xFFFF00000004LL,
    Mode2Form1Read = 0xFFFF00000005LL,
    Mode2Form2Read = 0xFFFF00000006LL,
    CDDA_Commands = 0xFFFF00000007LL,
    CDDA_StreamIsAccurate = 0xFFFF00000008LL,
    R_W_SubChannelsRead = 0xFFFF00000009LL,
    R_W_SubChannelsDeint = 0xFFFF0000000ALL,
    C2_Pointers = 0xFFFF0000000BLL,
    ISRC_Read = 0xFFFF0000000CLL,
    UPC_Read = 0xFFFF0000000DLL,
    BarcodeRead = 0xFFFF0000000ELL,
    LockMedia = 0xFFFF0000000FLL,
    LockState = 0xFFFF00000010LL,
    PreventJumper = 0xFFFF00000011LL,
    Eject = 0xFFFF00000012LL,
    SeparateVolumeLevels = 0xFFFF00000013LL,
    SeparateChannelMute = 0xFFFF00000014LL,
    ChangerDiscPresent = 0xFFFF00000015LL,
    ChangerSoftwareSlotSelection = 0xFFFF00000016LL,
    ChangerSideChangeCapable = 0xFFFF00000017LL,
    R_W_SubChannelsInLeadIn = 0xFFFF00000018LL,
    Method2AddressingFixedPackets = 0xFFFF00000019LL,
    CD_Text_Read = 0xFFFF0000001ALL,
    CD_Text_Write = 0xFFFF0000001BLL,
    DAO_Raw = 0xFFFF0000001CLL,
    DAO_16 = 0xFFFF0000001DLL,
    DAO_96_Pack = 0xFFFF0000001ELL,
    DAO_96_Raw = 0xFFFF0000001FLL,
    PacketWrite = 0xFFFF00000020LL,
    LabelFlash = 0xFFFF00000021LL,
    LightScribe = 0xFFFF00000022LL,
    ReadBlurayRXl = 0xFFFF00000023LL,
    ReadBlurayReXl = 0xFFFF00000024LL,
    WriteBlurayRXl = 0xFFFF00000025LL,
    WriteBlurayReXl = 0xFFFF00000026LL
  };
```

C#

```
  public enum Capabilities : System::Int64 {
    ReadCdR = 0x00000000001LL,
    ReadCdRw = 0x00000000002LL,
    ReadDvd = 0x00000000004LL,
    ReadDvdR = 0x00000000008LL,
    ReadDvdRw = 0x00000000010LL,
    ReadDvdRam = 0x00000000020LL,
    ReadDvdRPlus = 0x00000000040LL,
    ReadDvdRwPlus = 0x00000000080LL,
    ReadDvdDL = 0x00000000100LL,
    ReadDvdMrDL = 0x00000000200LL,
    ReadBlurayR = 0x00000000400LL,
    ReadBlurayRe = 0x00000000800LL,
    ReadBlurayRom = 0x00000001000LL,
    ReadHdDvdR = 0x00000002000LL,
    ReadHdDvdRw = 0x00000004000LL,
```

```
    ReadHdDvdRom = 0x00000008000LL,
    ReadMountRainer = 0x00000020000LL,
    ReadCdRwCav = 0x00000040000LL,
    WriteCdR = 0x00000080000LL,
    WriteCdRw = 0x00000100000LL,
    WriteDvdR = 0x00000200000LL,
    WriteDvdRw = 0x00000400000LL,
    WriteDvdRam = 0x00000800000LL,
    WriteDvdRPlus = 0x00001000000LL,
    WriteDvdRwPlus = 0x00002000000LL,
    WriteDvdDL = 0x00004000000LL,
    WriteDvdMrDL = 0x00008000000LL,
    WriteBlurayR = 0x00010000000LL,
    WriteBlurayRe = 0x00020000000LL,
    WriteHdDvdR = 0x00040000000LL,
    WriteHdDvdRw = 0x00080000000LL,
    WriteMountRainer = 0x00400000000LL,
    WriteCdRwCav = 0x00800000000LL,
    WriteTest = 0x01000000000LL,
    UnderrunProtection = 0x02000000000LL,
    Smart = 0x04000000000LL,
    Multisession = 0x08000000000LL,
    CprmAuth = 0x10000000000LL,
    DefectManagement = 0x20000000000LL,
    Streaming = 0x40000000000LL,
    ReadDvdRDLPlus = 0x080000000000LL,
    ReadDvdRwDLPlus = 0x100000000000LL,
    WriteDvdRDLPlus = 0x200000000000LL,
    WriteDvdRwDLPlus = 0x400000000000LL,
    LayerJumpRecording = 0x800000000000LL,
    AnalogAudioPlayback = 0xFFFF00000001LL,
    CompositeAudioAndVideo = 0xFFFF00000002LL,
    DigitalPort1 = 0xFFFF00000003LL,
    DigitalPort2 = 0xFFFF00000004LL,
    Mode2Form1Read = 0xFFFF00000005LL,
    Mode2Form2Read = 0xFFFF00000006LL,
    CDDA_Commands = 0xFFFF00000007LL,
    CDDA_StreamIsAccurate = 0xFFFF00000008LL,
    R_W_SubChannelsRead = 0xFFFF00000009LL,
    R_W_SubChannelsDeint = 0xFFFF0000000ALL,
    C2_Pointers = 0xFFFF0000000BLL,
    ISRC_Read = 0xFFFF0000000CLL,
    UPC_Read = 0xFFFF0000000DLL,
    BarcodeRead = 0xFFFF0000000ELL,
    LockMedia = 0xFFFF0000000FLL,
    LockState = 0xFFFF00000010LL,
    PreventJumper = 0xFFFF00000011LL,
    Eject = 0xFFFF00000012LL,
    SeparateVolumeLevels = 0xFFFF00000013LL,
    SeparateChannelMute = 0xFFFF00000014LL,
    ChangerDiscPresent = 0xFFFF00000015LL,
    ChangerSoftwareSlotSelection = 0xFFFF00000016LL,
    ChangerSideChangeCapable = 0xFFFF00000017LL,
    R_W_SubChannelsInLeadIn = 0xFFFF00000018LL,
    Method2AddressingFixedPackets = 0xFFFF00000019LL,
    CD_Text_Read = 0xFFFF0000001ALL,
    CD_Text_Write = 0xFFFF0000001BLL,
    DAO_Raw = 0xFFFF0000001CLL,
    DAO_16 = 0xFFFF0000001DLL,
    DAO_96_Pack = 0xFFFF0000001ELL,
    DAO_96_Raw = 0xFFFF0000001FLL,
    PacketWrite = 0xFFFF00000020LL,
    LabelFlash = 0xFFFF00000021LL,
    LightScribe = 0xFFFF00000022LL,
    ReadBlurayRXl = 0xFFFF00000023LL,
    ReadBlurayReXl = 0xFFFF00000024LL,
    WriteBlurayRXl = 0xFFFF00000025LL,
    WriteBlurayReXl = 0xFFFF00000026LL
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
ReadCdR = 0x00000000001LL	Drive Feature: Device is able to read CD-R.
ReadCdRw = 0x00000000002LL	Drive Feature: Device is able to read CD ReWriteable.
ReadDvd = 0x00000000004LL	Drive Feature: Device is able to read DVD-R.
ReadDvdR = 0x00000000008LL	Drive Feature: Device is able to read DVD-R.
ReadDvdRw = 0x00000000010LL	Drive Feature: Device is able to read DVD-ReWriteable.
ReadDvdRam = 0x00000000020LL	Drive Feature: Device is able to read DVD-RAM.
ReadDvdRPlus = 0x00000000040LL	Drive Feature: Device is able to read DVD+R.
ReadDvdRwPlus = 0x00000000080LL	Drive Feature: Device is able to read DVD+ReWriteable.
ReadDvdDL = 0x00000000100LL	Drive Feature: Device is able to read DVD-R Double Layer.
ReadDvdMrDL = 0x00000000200LL	Drive Feature: Device is able to read DVD-R Double Layer.
ReadBlurayR = 0x00000000400LL	Drive Feature: Device is able to read Blu-Ray R.
ReadBlurayRe = 0x00000000800LL	Drive Feature: Device is able to read Blu-Ray ReWriteable.
ReadBlurayRom = 0x00000001000LL	Drive Feature: Device is able to read Blu-Ray ROM.
ReadHdDvdR = 0x00000002000LL	Drive Feature: Device is able to read HDDVD R.
ReadHdDvdRw = 0x00000004000LL	Drive Feature: Device is able to read HDDVD ReWriteable.
ReadHdDvdRom = 0x00000008000LL	Drive Feature: Device is able to read HDDVD Rom.
ReadMountRainer = 0x00000020000LL	Drive Feature: Device is able to read Mount Rainer.
ReadCdRwCav = 0x00000040000LL	Drive Feature: Device is able to read CD-RW media that is designed for CAV recording.
WriteCdR = 0x00000080000LL	Drive Feature: Device is able to write CD-Recordable.
WriteCdRw = 0x00000100000LL	Drive Feature: Device is able to write CD-ReWriteable.
WriteDvdR = 0x00000200000LL	Drive Feature: Device is able to write DVD-R.
WriteDvdRw = 0x00000400000LL	Drive Feature: Device is able to write DVD-ReWriteable.
WriteDvdRam = 0x00000800000LL	Drive Feature: Device is able to write DVD-RAM.
WriteDvdRPlus = 0x00001000000LL	Drive feature: Device is able to write DVD+R.
WriteDvdRwPlus = 0x00002000000LL	Drive feature: Device is able to write DVD+ReWriteable.
WriteDvdDL = 0x00004000000LL	Drive Feature: Device is able to write DVD-R Double Layer.
WriteDvdMrDL = 0x00008000000LL	Drive Feature: Device is able to write DVD-R Double Layer.
WriteBlurayR = 0x00010000000LL	Drive Feature: Device is able to write Blu-Ray Recordable.
WriteBlurayRe = 0x00020000000LL	Drive Feature: Device is able to write Blu-Ray ReWriteable.
WriteHdDvdR = 0x00040000000LL	Drive Feature: Device is able to write HDDVD Reordable.
WriteHdDvdRw = 0x00080000000LL	Drive Feature: Device is able to write HDDVD ReWriteable.
WriteMountRainer = 0x00400000000LL	Drive Feature: Device is able to write Mount Rainer.
WriteCdRwCav = 0x00800000000LL	Drive Feature: Device is able to read CD-RW media that is designed for CAV recording.
WriteTest = 0x01000000000LL	Drive feature: Device is able to simulate burning.
UnderrunProtection = 0x02000000000LL	Drive Feature: Device is able to perform Buffer protection functions like Burn-Proof.
Smart = 0x04000000000LL	Drive Feature: Device is able to perform Self-Monitoring Analysis and Reporting Technology.
Multisession = 0x08000000000LL	Drive Feature: Device is able to perform multi-session media.
CprmAuth = 0x10000000000LL	Drive Feature: Device is able to CPRM authentification and key management.
DefectManagement = 0x20000000000LL	Drive Feature: Device has defect management feature. Available to provide defect-free address space. For example BD recorders provide this feature.

Streaming = 0x40000000000LL	Drive Feature: Device is able to perform reading and writing within specified performance ranges.
ReadDvdRDLPlus = 0x080000000000LL	Drive feature: Device is able to read DVD+R Double Layer.
ReadDvdRwDLPlus = 0x100000000000LL	Drive feature: Device is able to read DVD+ReWriteable Double Layer.
WriteDvdRDLPlus = 0x200000000000LL	Drive feature: Device is able to write DVD+R Double Layer.
WriteDvdRwDLPlus = 0x400000000000LL	Drive feature: Device is able to write DVD+ReWriteable Double Layer.
LayerJumpRecording = 0x800000000000LL	Drive feature: Device is able to perform Layer Jump Recording.
AnalogAudioPlayback = 0xFFFF00000001LL	Drive feature: Device support playback analog audio.
CompositeAudioAndVideo = 0xFFFF00000002LL	Drive feature: The Logical Unit shall be capable of delivering a composite Audio and Video data stream from an independent digital port.
DigitalPort1 = 0xFFFF00000003LL	Drive feature: The Logical Unit shall support digital output (IEC958) on port 1.
DigitalPort2 = 0xFFFF00000004LL	Drive feature: The Logical Unit shall support digital output (IEC958) on port 2.
Mode2Form1Read = 0xFFFF00000005LL	Drive feature: Device is able to read disks that are recorded with Mode2 Form1 format.
Mode2Form2Read = 0xFFFF00000006LL	Drive feature: Device is able to read disks that are recorded with Mode2 Form2 format.
CDDA_Commands = 0xFFFF00000007LL	Drive feature: Device is able to work with CDDA command set.
CDDA_StreamIsAccurate = 0xFFFF00000008LL	Drive feature: The Logical Unit shall support an audio location without losing place to continue the READ CD command.
R_W_SubChannelsRead = 0xFFFF00000009LL	Drive feature: The Logical Unit shall support reading R-W sub-channel via the READ CD command.
R_W_SubChannelsDeint = 0xFFFF0000000ALL	Drive feature: The Logical Unit shall support reading R-W sub-channel via the READ CD command with the returned data de-interleaved and error corrected
C2_Pointers = 0xFFFF0000000BLL	Drive feature: Device support reading C2 error pointers and C2 block error flags.
ISRC_Read = 0xFFFF0000000CLL	Drive feature: Device is able to read the ISRC (International Standard Recording Code) info of a disk.
UPC_Read = 0xFFFF0000000DLL	Drive feature: Device is able to read the UPC info of a disk.
BarcodeRead = 0xFFFF0000000ELL	Drive feature: Device is able to read the disk barcode information.
LockMedia = 0xFFFF0000000FLL	Drive feature: Device is able to lock the media.
LockState = 0xFFFF00000010LL	Drive feature: Device is able return the lock state.
PreventJumper = 0xFFFF00000011LL	Drive feature: The Logical Unit has a physical jumper named the Prevent/Allow Jumper and the jumper is present.
Eject = 0xFFFF00000012LL	Drive feature: The Logical Unit shall support media eject via the START STOP UNIT command with the LoEj bit set.
SeparateVolumeLevels = 0xFFFF00000013LL	Drive feature: The Logical Unit shall support separately controllable audio levels for each supported channel via the CD Audio Control Page.
SeparateChannelMute = 0xFFFF00000014LL	Drive feature: The Logical Unit shall support independently muting each audio channel via the CD Audio Control Page.
ChangerDiscPresent = 0xFFFF00000015LL	Drive feature: The Logical Unit contains an embedded changer, and after a reset condition or a magazine change, the Logical Unit is capable of reporting the exact contents of the slots.

ChangerSoftwareSlotSelection = 0xFFFF00000016LL	Drive feature: This bit controls the behavior of the LOAD/UNLOAD MEDIUM command when trying to load a Slot with no Disc present.
ChangerSideChangeCapable = 0xFFFF00000017LL	Drive feature: The Logical Unit is capable of selecting both sides of Discs.
R_W_SubChannelsInLeadIn = 0xFFFF00000018LL	Drive feature: The Logical Unit is capable of reading the raw R-W Sub-channel information from the Lead-in.
Method2AddressingFixedPackets = 0xFFFF00000019LL	Drive feature: Device support reading fixed packet tracks on CD-R/RW media where the Addressing type is Method 2.
CD_Text_Read = 0xFFFF0000001ALL	Drive feature: Device is able to read the CD-TEXT infromation of a disk (sub channel).
CD_Text_Write = 0xFFFF0000001BLL	Drive feature: Device is able to write the CD-TEXT infromation of a disk (sub channel).
DAO_Raw = 0xFFFF0000001CLL	Drive feature: Device can write DAO raw write method.
DAO_16 = 0xFFFF0000001DLL	Drive feature: Device can write DAO 16 method.
DAO_96_Pack = 0xFFFF0000001ELL	Drive feature: Device can write DAO 96 packet write method.
DAO_96_Raw = 0xFFFF0000001FLL	Drive feature: Device can write DAO 96 raw write method.
PacketWrite = 0xFFFF00000020LL	Drive feature: The drive can perform packet writing.
LabelFlash = 0xFFFF00000021LL	Drive feature: Device is able to write Labelflash discs.
LightScribe = 0xFFFF00000022LL	Drive feature: Device is able to write Lightscribe discs.
ReadBlurayRXl = 0xFFFF00000023LL	Drive feature: Device is able to read BD-R XL.
ReadBlurayReXl = 0xFFFF00000024LL	Drive feature: Device is able to read BD-RE XL.
WriteBlurayRXl = 0xFFFF00000025LL	Drive feature: Device is able to write BD-R XL.
WriteBlurayReXl = 0xFFFF00000026LL	Drive feature: Device is able to write BD-RE XL.

Description

This enumeration defines the capabilities of the device. Please click on the feature you want to get information for.

1.1.2.10 IsoSDK::CDTextContentItem Enumeration

C++

```
enum class CDTextContentItem {
  Arrager = 0,
  Composer = 1,
  Title = 2,
  Message = 3,
  Performer = 4,
  SongWriter = 5
};
```

C#

```
public enum CDTextContentItem {
  Arrager = 0,
  Composer = 1,
  Title = 2,
  Message = 3,
  Performer = 4,
  SongWriter = 5
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Arrager = 0	CD-Text Content Items: Arranger field.
Composer = 1	CD-Text Content Items: Composer field.

Title = 2	CD-Text Content Items: Title field.
Message = 3	CD-Text Content Items: Message field.
Performer = 4	CD-Text Content Items: Performer field.
SongWriter = 5	CD-Text Content Items: Songwriter field.

Description

This enumeration defines the types of CD-Text information the IsoSDK (see page 1) supports.

1.1.2.11 IsoSDK::CreateImageParams Structure

C++

```
ref struct CreateImageParams {
  String ^ ImagePath;
  String ^ BadSectorsFilePath;
  ImageFormat ImageType;
  int VerifyBufferSectors;
  bool FullCapacity;
  ReadErrorCorrectionParams ^ ErrorParams;
};
```

C#

```
public struct CreateImageParams {
  public String ImagePath;
  public String BadSectorsFilePath;
  public ImageFormat ImageType;
  public int VerifyBufferSectors;
  public bool FullCapacity;
  public ReadErrorCorrectionParams ErrorParams;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ ImagePath;	The path and file name of the target disk image.
String ^ BadSectorsFilePath;	If you fill this parameter with a path and filename, the IsoSDK (see page 1) will create a text file with a list of bad sectors.
ImageFormat ImageType;	The type of disk image according to the ImageFormat (see page 186) enumeration.
int VerifyBufferSectors;	Set the VerifyBufferSectors while verify the copy. Valid value 1-27. Other values will cause an error. The buffer size is not dedicated to the hardware. It is the amount of sectors the IsoSDK (see page 1) will read for each verify step.
bool FullCapacity;	Set to true to read the full disk, all sectors. When set to false, the IsoSDK (see page 1) will read the track size. If a disc was formatted it is filled with 0 sectrors so in formatted disc the IsoDK will also read the 0 Sectors up to disc size. Audio, Mixed Mode and VideoCD always set to false because they may have more than one track.
ReadErrorCorrectionParams ^ ErrorParams;	A structure ReadErrorCorrectionParams (see page 198).

Description

A structure that contains information for the ImageCreate operation.

1.1.2.12 IsoSDK::DeviceIndex Enumeration

C++

```
enum class DeviceIndex {
  Current = BS_CURRENT_DEVICE,
  Write = BS_BURN_DEVICE,
  Read = BS_READ_DEVICE
};
```

C#

```
public enum DeviceIndex {
  Current = BS_CURRENT_DEVICE,
  Write = BS_BURN_DEVICE,
  Read = BS_READ_DEVICE
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Current = BS_CURRENT_DEVICE	IsoSDK (see page 1) device type: Default device index
Write = BS_BURN_DEVICE	IsoSDK (see page 1) device type: Burn Device
Read = BS_READ_DEVICE	IsoSDK (see page 1) device type: Read Device

Description

This enumeration defines the possible selection for devices.

1.1.2.13 IsoSDK::DeviceInformation Structure

C++

```
ref struct DeviceInformation {
  String ^ VendorID;
  String ^ ProductID;
  String ^ ProductRevision;
};
```

C#

```
public struct DeviceInformation {
  public String VendorID;
  public String ProductID;
  public String ProductRevision;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ VendorID;	Vendor ID of the current device.
String ^ ProductID;	Product ID of the current device.
String ^ ProductRevision;	Product revision of the current device.

Description

A structure that contains device information of a device from the internal device list.

1.1.2.14 IsoSDK::DiscSignature Enumeration new

C++

```
enum class DiscSignature {
  Encrypted = 1,
  Compressed = 2,
  EncryptedCompressed = 3
};
```

C#

```
public enum DiscSignature {
  Encrypted = 1,
  Compressed = 2,
  EncryptedCompressed = 3
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Encrypted = 1	Disc or Image is encrypted
Compressed = 2	Disc or Image is compressed
EncryptedCompressed = 3	Disc or Image is compressed and encrypted

Description

This enumeration defines the compression and/or encryption state of the disc or image.

1.1.2.15 IsoSDK::DiskCopyOptions Structure

C++

```
ref struct DiskCopyOptions {
  WriteMethod WriteMethod;
  ReadMode ReadMode;
  bool VerifyAfterBurn;
  ReadErrorCorrectionParams ^ ErrorParams;
  bool EjectAfterBurn;
  int VerifyBufferSectors;
  bool FullCapacity;
};
```

C#

```
public struct DiskCopyOptions {
  public WriteMethod WriteMethod;
  public ReadMode ReadMode;
  public bool VerifyAfterBurn;
  public ReadErrorCorrectionParams ErrorParams;
  public bool EjectAfterBurn;
  public int VerifyBufferSectors;
  public bool FullCapacity;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
WriteMethod WriteMethod;	The write method for the target device according to the WriteMethod enumeration.

ReadMode ReadMode;	The read mode of the source disk according to the ReadMode enumeration.
bool VerifyAfterBurn;	Indicates to verify (true) the target disk after burning.
ReadErrorCorrectionParams ^ ErrorParams;	A structure ReadErrorCorrectionParams (see page 198).
bool EjectAfterBurn;	This argument determines whether the medium will be ejected after the burning process. true ejects the medium, and false prevents this.
int VerifyBufferSectors;	Set the VerifyBufferSize while verify the copy. Valid value 1-27. Other values will cause an error.
bool FullCapacity;	Set to true to read the full disk. When set to false, the IsoSDK (see page 1) will read the track size. Audio, Mixed Mode and VideoCD always set to false.

Description

A structure that contains information for the disk copy operation.

1.1.2.16 IsoSDK::ErrorCode Enumeration

C++

```
enum class ErrorCode {
  SdkErrorNo = 0,
  SdkErrorNotAllowed = 1,
  ScsiErrorParam10 = 2,
  SdkErrorBadRequest = 3,
  SdkErrorGeneral = 4,
  ScsiErrorAspi01 = 5,
  ScsiErrorAspi06 = 6,
  SdkErrorInvalidPath = 7,
  SdkErrorInvalidSrcPath = 8,
  SdkErrorInvalidDestPath = 9,
  SdkErrorInvalidFileName = 10,
  SdkErrorPathExists = 11,
  SdkErrorFileExists = 12,
  SdkCueErrorCommand06 = 13,
  SdkErrorInvalidFileFormat = 15,
  SdkErrorFileOpen = 16,
  SdkErrorCorruptOrInvalidCueFile = 17,
  SdkErrorBinFileNotFound = 18,
  SdkErrorNotImplemented = 19,
  SdkErrorNotAllowedForThisBurner = 20,
  ScsiErrorDisk29 = 21,
  ScsiErrorDisk30 = 22,
  ScsiErrorDisk31 = 23,
  ScsiErrorDisk32 = 24,
  SdkErrorUnknownTextid = 25,
  SdkErrorMoreSpaceNeeded = 26,
  SdkErrorBurnInProgress = 27,
  SdkErrorInvalidSessionNumber = 28,
  SdkErrorEmptyPassword = 29,
  SdkErrorUnknown = 101,
  ScsiErrorAspi02 = 102,
  ScsiErrorAspi03 = 103,
  SdkErrorInvalidsrb = 104,
  SdkErrorBufferalignment = 105,
  ScsiErrorAspi04 = 106,
  SdkErrorBuffertoobig = 107,
  SdkErrorTimeout = 108,
  SdkErrorSrbtimeout = 109,
  SdkErrorMessagereject = 110,
  ScsiErrorAspi05 = 111,
  SdkErrorParityerr = 112,
  ScsiError107 = 113,
  SdkErrorSelectiontimeout = 114,
  SdkErrorDataoverrun = 115,
```

```
SdkErrorUnexpectedbusfree = 116,
SdkErrorCheckcondition = 117,
ScsiErrorAspi08 = 118,
ScsiErrorAspi09 = 119,
ScsiErrorAspi10 = 120,
SdkErrorDatarecovererror = 121,
SdkErrorNotready = 122,
ScsiErrorDisk33 = 123,
SdkErrorHardwareerror = 124,
SdkErrorUnitattention = 126,
SdkErrorDataprotect = 127,
SdkErrorErasecheck = 128,
SdkErrorCopyaborted = 129,
SdkErrorAbortedcommand = 130,
SdkErrorVolumeoverflow = 131,
SdkErrorMiscompare = 132,
SdkErrorReserved = 133,
SdkErrorFilemark = 134,
SdkErrorIllegallength = 136,
SdkErrorIncorrectlength = 137,
SdkErrorAborted = 138,
SdkErrorFileinuse = 139,
SdkErrorCreatefile = 140,
ScsiErrorAspi07 = 141,
ScsiErrorWrite12 = 142,
SdkErrorNotsupported = 143,
SdkErrorNextaddress = 144,
SdkErrorTooMuchData = 145,
SdkErrorMaxdirs = 146,
SdkErrorMaxfiles = 147,
SdkErrorErrinvalidfilename = 148,
SdkErrorImportsession = 149,
SdkErrorIsoimagenotfound = 150,
SdkIntError1 = 151,
SdkIntError2 = 152,
SdkIntError3 = 153,
SdkIntError4 = 154,
SdkIntError5 = 155,
SdkCueErrorSendingCue = 156,
SdkCueErrorUeol = 157,
SdkCueErrorField = 158,
SdkCueErrorFile = 159,
SdkCueErrorCommand01 = 160,
SdkIntErrorFormat = 161,
SdkErrorNotAllowedForThisUdfVersion = 162,
SdkErrorInvalidUdfVersion = 163,
SdkErrorMp3libNotFound = 164,
SdkErrorIncompatibleFsType = 165,
SdkErrorCompressionConflict = 166,
SdkErrorEncryptionConflict = 167,
SdkErrorInvalidDirIndex = 168,
SdkErrorInvalidMcn = 169,
SdkErrorInvalidIsrc = 170,
IleTooLongDirectoryNesting = 171,
IleTooBigFile = 172,
SdkErrorInvalidIndex = 173,
SdkErrorTooMuchIndexes = 174,
SdkErrorEnclibNotFound = 175,
SdkErrorCdTextNotFound = 176,
SdkErrorNetTagsDisk = 177,
SdkErrorNetTagsConnect = 178,
SdkErrorNetTagsServer = 179,
SdkErrorNetTagsNoMatch = 180,
SdkErrorNetTagsInternal = 181,
ScsiError01 = 200,
ScsiError02 = 201,
ScsiErrorAudio01 = 202,
ScsiErrorAudio02 = 203,
ScsiErrorAudio03 = 204,
ScsiErrorAudio04 = 205,
ScsiErrorAudio05 = 206,
ScsiError001 = 207,
```

```
ScsiError002 = 208,
ScsiErrorUnit01 = 209,
ScsiErrorUnit02 = 210,
ScsiErrorUnit03 = 211,
ScsiErrorUnit04 = 212,
ScsiErrorUnit05 = 213,
ScsiErrorUnit06 = 214,
ScsiErrorUnit07 = 215,
ScsiErrorUnit08 = 216,
ScsiErrorUnit09 = 217,
ScsiErrorUnit10 = 218,
ScsiError004 = 219,
ScsiError005 = 220,
ScsiErrorUnit11 = 221,
ScsiErrorUnit12 = 222,
ScsiErrorUnit13 = 223,
ScsiErrorUnit14 = 224,
ScsiError006 = 225,
ScsiError007 = 226,
ScsiError008 = 227,
ScsiError009 = 228,
ScsiError010 = 229,
ScsiError011 = 230,
ScsiErrorLog01 = 231,
ScsiErrorAtt01 = 232,
ScsiErrorAtt02 = 233,
ScsiErrorAtt03 = 234,
ScsiErrorWrite01 = 235,
ScsiErrorWrite02 = 236,
ScsiErrorWrite03 = 237,
ScsiErrorWrite04 = 238,
ScsiErrorWrite05 = 239,
ScsiErrorExt01 = 240,
ScsiErrorExt02 = 241,
ScsiErrorTarget01 = 242,
ScsiErrorTarget02 = 243,
ScsiErrorTarget03 = 244,
ScsiErrorTarget04 = 245,
ScsiErrorRead01 = 246,
ScsiErrorRead02 = 247,
ScsiError012 = 248,
ScsiError013 = 249,
ScsiErrorCirc01 = 250,
ScsiErrorCrc01 = 251,
ScsiErrorDecom01 = 252,
ScsiErrorDrive01 = 253,
ScsiErrorDrive02 = 254,
ScsiError014 = 255,
ScsiError015 = 256,
ScsiError016 = 257,
ScsiError017 = 258,
ScsiErrorMech01 = 259,
ScsiErrorMech02 = 260,
ScsiErrorRecover01 = 261,
ScsiErrorRecover02 = 262,
ScsiErrorRecover03 = 263,
ScsiErrorRecover04 = 264,
ScsiErrorRecover05 = 265,
ScsiErrorRecover06 = 266,
ScsiErrorRecover07 = 267,
ScsiErrorRecover08 = 268,
ScsiErrorRecover09 = 269,
ScsiErrorRecover10 = 270,
ScsiErrorRecover11 = 271,
ScsiErrorRecover12 = 272,
ScsiErrorRecover13 = 273,
ScsiErrorRecover14 = 274,
ScsiErrorRecover15 = 275,
ScsiErrorRecover16 = 276,
ScsiErrorRecover17 = 277,
ScsiError018 = 278,
ScsiError019 = 279,
```

```
ScsiError020 = 280,
ScsiError021 = 281,
ScsiError022 = 282,
ScsiError023 = 283,
ScsiError024 = 284,
ScsiErrorCdb01 = 285,
ScsiErrorCdb02 = 286,
ScsiErrorUnit16 = 287,
ScsiErrorParam01 = 288,
ScsiErrorParam02 = 289,
ScsiErrorParam03 = 290,
ScsiErrorParam04 = 291,
ScsiError025 = 292,
ScsiError026 = 293,
ScsiErrorSegm01 = 296,
ScsiErrorSegm02 = 297,
ScsiErrorSegm03 = 298,
ScsiError027 = 299,
ScsiError028 = 300,
ScsiErrorSegm04 = 301,
ScsiErrorWrite06 = 302,
ScsiErrorWrite07 = 303,
ScsiErrorWrite08 = 305,
ScsiErrorWrite09 = 306,
ScsiErrorWrite10 = 307,
ScsiErrorWrite11 = 308,
ScsiError029 = 309,
ScsiError030 = 310,
ScsiError031 = 311,
ScsiError032 = 312,
ScsiError033 = 313,
ScsiError034 = 314,
ScsiError035 = 315,
ScsiError036 = 316,
ScsiError037 = 317,
ScsiErrorParam05 = 318,
ScsiErrorParam06 = 319,
ScsiErrorParam07 = 320,
ScsiError038 = 321,
ScsiError039 = 322,
ScsiError040 = 323,
ScsiError041 = 324,
ScsiErrorCommand02 = 325,
ScsiError042 = 326,
ScsiError043 = 327,
ScsiError044 = 328,
ScsiError045 = 329,
ScsiErrorCommand03 = 330,
ScsiErrorDisk01 = 331,
ScsiErrorDisk02 = 332,
ScsiErrorDisk03 = 333,
ScsiErrorDisk04 = 334,
ScsiErrorDisk05 = 335,
ScsiErrorDisk06 = 336,
ScsiErrorDisk07 = 337,
ScsiErrorDisk08 = 338,
ScsiErrorDisk09 = 339,
ScsiErrorDisk10 = 340,
ScsiErrorDisk11 = 341,
ScsiErrorDisk12 = 342,
ScsiErrorDisk13 = 343,
ScsiError046 = 344,
ScsiError047 = 345,
ScsiError048 = 346,
ScsiError049 = 347,
ScsiError050 = 348,
ScsiError051 = 349,
ScsiError052 = 350,
ScsiErrorParam08 = 351,
ScsiErrorParam09 = 352,
ScsiErrorDisk14 = 353,
ScsiErrorDisk15 = 354,
```

```
ScsiErrorDisk16 = 355,
ScsiErrorDisk17 = 356,
ScsiErrorDisk18 = 357,
ScsiErrorDisk19 = 358,
ScsiErrorDisk20 = 359,
ScsiErrorDisk21 = 360,
ScsiErrorDisk22 = 361,
ScsiErrorDisk23 = 362,
ScsiErrorDisk24 = 363,
ScsiErrorDisk25 = 364,
ScsiErrorDisk26 = 365,
ScsiErrorMech03 = 366,
ScsiError053 = 367,
ScsiErrorUnit17 = 368,
ScsiErrorUnit18 = 369,
ScsiErrorUnit19 = 370,
ScsiErrorUnit20 = 371,
ScsiErrorUnit21 = 372,
ScsiError054 = 373,
ScsiError055 = 374,
ScsiError056 = 375,
ScsiError057 = 376,
ScsiError058 = 377,
ScsiError059 = 378,
ScsiError060 = 379,
ScsiError061 = 380,
ScsiError062 = 381,
ScsiError063 = 382,
ScsiErrorVol01 = 383,
ScsiErrorVol02 = 384,
ScsiErrorVol03 = 385,
ScsiErrorVol04 = 386,
ScsiError064 = 387,
ScsiError065 = 388,
ScsiErrorDisk27 = 389,
ScsiErrorDisk28 = 390,
ScsiError066 = 391,
ScsiError067 = 392,
ScsiError068 = 393,
ScsiError069 = 394,
ScsiError070 = 395,
ScsiError071 = 396,
ScsiError072 = 397,
ScsiError073 = 398,
ScsiError074 = 399,
ScsiError075 = 400,
ScsiError076 = 401,
ScsiErrorCommand04 = 402,
ScsiError077 = 403,
ScsiErrorUnit22 = 404,
ScsiErrorCommand05 = 405,
ScsiError078 = 410,
ScsiError079 = 411,
ScsiError080 = 412,
ScsiError081 = 413,
ScsiError082 = 414,
ScsiError083 = 415,
ScsiError084 = 416,
ScsiError085 = 417,
ScsiErrorLog02 = 418,
ScsiError086 = 419,
ScsiErrorLog03 = 420,
ScsiErrorLog04 = 421,
ScsiError087 = 422,
ScsiErrorUnit23 = 424,
ScsiError088 = 425,
ScsiError089 = 426,
ScsiError090 = 427,
ScsiError091 = 428,
ScsiError092 = 429,
ScsiError093 = 430,
ScsiError094 = 431,
```

```
ScsiError095 = 432,
ScsiError096 = 433,
ScsiError097 = 434,
ScsiError098 = 435,
ScsiError099 = 436,
ScsiErrorDcss01 = 437,
ScsiErrorDcss02 = 438,
ScsiErrorDcss03 = 439,
ScsiErrorDcss04 = 440,
ScsiErrorDcss05 = 441,
ScsiErrorDcss06 = 442,
ScsiErrorSession01 = 443,
ScsiErrorSession02 = 444,
ScsiErrorSession03 = 445,
ScsiErrorSession04 = 446,
ScsiError100 = 447,
ScsiError101 = 448,
ScsiError102 = 449,
ScsiError103 = 450,
ScsiError104 = 451,
ScsiError105 = 452,
ScsiErrorAlloc01 = 453,
ScsiErrorAlloc02 = 454,
ScsiError106 = 455,
ScsiErrorDisk34 = 456,
MultiError01 = 457,
MultiError02 = 458,
ScsiErrorDisk35 = 459,
ScsiError108 = 460,
ScsiError109 = 461,
ScsiErrorAtt04 = 462,
ScsiErrorAtt05 = 463,
ScsiError110 = 464,
ScsiError111 = 465,
ScsiError112 = 466,
ScsiErrorDisk36 = 467,
ScsiError113 = 468,
ScsiError114 = 469,
ScsiError115 = 470,
ScsiError116 = 471,
ScsiError117 = 472,
ScsiError118 = 473,
ScsiError119 = 474,
ScsiErrorDcss07 = 475,
ScsiErrorDcss08 = 476,
SdkMessageWait = 500,
SdkMessageWritestart = 501,
SdkMessageEreasestart = 502,
SdkMessageExtrFile = 503,
SdkMessageSimulate = 504,
SdkMessageImport = 505,
SdkMessageFormat = 506,
SdkMessageFormatDone = 507,
SdkMessageImagecreatestart = 508,
SdkMessage01 = 600,
SdkMessage02 = 601,
SdkMessage03 = 602,
SdkMessage04 = 603,
SdkMessage05 = 604,
SdkMessage06 = 605,
SdkMessage07 = 606,
SdkMessage08 = 607,
SdkMessage10 = 608,
SdkMessage11 = 609,
SdkMessage12 = 610,
SdkMessage13 = 611,
SdkMessage14 = 612,
SdkMessage15 = 613,
SdkMessage16 = 614,
GuiResource01 = 630,
GuiResource02 = 631,
GuiResource03 = 632,
```

```
    GuiResource04 = 633,
    GuiResource05 = 634,
    GuiResource06 = 635,
    GuiResource07 = 636,
    GuiResource08 = 637,
    GuiResource09 = 638,
    GuiResource10 = 639,
    GuiResource11 = 640,
    GuiResource12 = 641,
    GuiResource13 = 642,
    GuiResource14 = 643,
    GuiResource15 = 644,
    GuiResource16 = 645,
    GuiResource17 = 646,
    GuiResource18 = 647,
    GuiResource19 = 648,
    GuiResource20 = 649,
    GuiResource21 = 650,
    GuiResource22 = 651,
    GuiResource23 = 652,
    GuiResource24 = 653,
    GuiResource25 = 654,
    GuiResource26 = 655,
    GuiResource27 = 656,
    GuiResource28 = 657,
    GuiResource29 = 658,
    GuiResource30 = 659,
    GuiResource31 = 660,
    GuiResource32 = 661,
    GuiResource33 = 662,
    GuiResource34 = 663,
    GuiResource35 = 664,
    GuiResource36 = 665,
    GuiResource37 = 666,
    GuiResource38 = 667,
    GuiResource39 = 668,
    GuiResource40 = 669,
    GuiResource41 = 670,
    GuiResource42 = 671,
    GuiResource43 = 672,
    GuiResource44 = 673,
    GuiResource45 = 674,
    GuiResource46 = 675,
    GuiError01 = 676,
    GuiError02 = 677,
    GuiResource47 = 678,
    GuiResource48 = 679,
    GuiResource49 = 680,
    GuiResource50 = 681,
    SdkVerifyErrorHddfileunreadable = 700,
    SdkVerifyErrorCdfileunreadable = 701,
    SdkVerifyErrorFilesdifferent = 702
  };
```

C#

```
  public enum ErrorCode {
    SdkErrorNo = 0,
    SdkErrorNotAllowed = 1,
    ScsiErrorParam10 = 2,
    SdkErrorBadRequest = 3,
    SdkErrorGeneral = 4,
    ScsiErrorAspi01 = 5,
    ScsiErrorAspi06 = 6,
    SdkErrorInvalidPath = 7,
    SdkErrorInvalidSrcPath = 8,
    SdkErrorInvalidDestPath = 9,
    SdkErrorInvalidFileName = 10,
    SdkErrorPathExists = 11,
    SdkErrorFileExists = 12,
    SdkCueErrorCommand06 = 13,
    SdkErrorInvalidFileFormat = 15,
```

```
SdkErrorFileOpen = 16,
SdkErrorCorruptOrInvalidCueFile = 17,
SdkErrorBinFileNotFound = 18,
SdkErrorNotImplemented = 19,
SdkErrorNotAllowedForThisBurner = 20,
ScsiErrorDisk29 = 21,
ScsiErrorDisk30 = 22,
ScsiErrorDisk31 = 23,
ScsiErrorDisk32 = 24,
SdkErrorUnknownTextid = 25,
SdkErrorMoreSpaceNeeded = 26,
SdkErrorBurnInProgress = 27,
SdkErrorInvalidSessionNumber = 28,
SdkErrorEmptyPassword = 29,
SdkErrorUnknown = 101,
ScsiErrorAspi02 = 102,
ScsiErrorAspi03 = 103,
SdkErrorInvalidsrb = 104,
SdkErrorBufferalignment = 105,
ScsiErrorAspi04 = 106,
SdkErrorBuffertoobig = 107,
SdkErrorTimeout = 108,
SdkErrorSrbtimeout = 109,
SdkErrorMessagereject = 110,
ScsiErrorAspi05 = 111,
SdkErrorParityerr = 112,
ScsiError107 = 113,
SdkErrorSelectiontimeout = 114,
SdkErrorDataoverrun = 115,
SdkErrorUnexpectedbusfree = 116,
SdkErrorCheckcondition = 117,
ScsiErrorAspi08 = 118,
ScsiErrorAspi09 = 119,
ScsiErrorAspi10 = 120,
SdkErrorDatarecovererror = 121,
SdkErrorNotready = 122,
ScsiErrorDisk33 = 123,
SdkErrorHardwareerror = 124,
SdkErrorUnitattention = 126,
SdkErrorDataprotect = 127,
SdkErrorErasecheck = 128,
SdkErrorCopyaborted = 129,
SdkErrorAbortedcommand = 130,
SdkErrorVolumeoverflow = 131,
SdkErrorMiscompare = 132,
SdkErrorReserved = 133,
SdkErrorFilemark = 134,
SdkErrorIllegallength = 136,
SdkErrorIncorrectlength = 137,
SdkErrorAborted = 138,
SdkErrorFileinuse = 139,
SdkErrorCreatefile = 140,
ScsiErrorAspi07 = 141,
ScsiErrorWrite12 = 142,
SdkErrorNotsupported = 143,
SdkErrorNextaddress = 144,
SdkErrorTooMuchData = 145,
SdkErrorMaxdirs = 146,
SdkErrorMaxfiles = 147,
SdkErrorErrinvalidfilename = 148,
SdkErrorImportsession = 149,
SdkErrorIsoimagenotfound = 150,
SdkIntError1 = 151,
SdkIntError2 = 152,
SdkIntError3 = 153,
SdkIntError4 = 154,
SdkIntError5 = 155,
SdkCueErrorSendingCue = 156,
SdkCueErrorUeol = 157,
SdkCueErrorField = 158,
SdkCueErrorFile = 159,
SdkCueErrorCommand01 = 160,
```

```
SdkIntErrorFormat = 161,
SdkErrorNotAllowedForThisUdfVersion = 162,
SdkErrorInvalidUdfVersion = 163,
SdkErrorMp3libNotFound = 164,
SdkErrorIncompatibleFsType = 165,
SdkErrorCompressionConflict = 166,
SdkErrorEncryptionConflict = 167,
SdkErrorInvalidDirIndex = 168,
SdkErrorInvalidMcn = 169,
SdkErrorInvalidIsrc = 170,
IleTooLongDirectoryNesting = 171,
IleTooBigFile = 172,
SdkErrorInvalidIndex = 173,
SdkErrorTooMuchIndexes = 174,
SdkErrorEnclibNotFound = 175,
SdkErrorCdTextNotFound = 176,
SdkErrorNetTagsDisk = 177,
SdkErrorNetTagsConnect = 178,
SdkErrorNetTagsServer = 179,
SdkErrorNetTagsNoMatch = 180,
SdkErrorNetTagsInternal = 181,
ScsiError01 = 200,
ScsiError02 = 201,
ScsiErrorAudio01 = 202,
ScsiErrorAudio02 = 203,
ScsiErrorAudio03 = 204,
ScsiErrorAudio04 = 205,
ScsiErrorAudio05 = 206,
ScsiError001 = 207,
ScsiError002 = 208,
ScsiErrorUnit01 = 209,
ScsiErrorUnit02 = 210,
ScsiErrorUnit03 = 211,
ScsiErrorUnit04 = 212,
ScsiErrorUnit05 = 213,
ScsiErrorUnit06 = 214,
ScsiErrorUnit07 = 215,
ScsiErrorUnit08 = 216,
ScsiErrorUnit09 = 217,
ScsiErrorUnit10 = 218,
ScsiError004 = 219,
ScsiError005 = 220,
ScsiErrorUnit11 = 221,
ScsiErrorUnit12 = 222,
ScsiErrorUnit13 = 223,
ScsiErrorUnit14 = 224,
ScsiError006 = 225,
ScsiError007 = 226,
ScsiError008 = 227,
ScsiError009 = 228,
ScsiError010 = 229,
ScsiError011 = 230,
ScsiErrorLog01 = 231,
ScsiErrorAtt01 = 232,
ScsiErrorAtt02 = 233,
ScsiErrorAtt03 = 234,
ScsiErrorWrite01 = 235,
ScsiErrorWrite02 = 236,
ScsiErrorWrite03 = 237,
ScsiErrorWrite04 = 238,
ScsiErrorWrite05 = 239,
ScsiErrorExt01 = 240,
ScsiErrorExt02 = 241,
ScsiErrorTarget01 = 242,
ScsiErrorTarget02 = 243,
ScsiErrorTarget03 = 244,
ScsiErrorTarget04 = 245,
ScsiErrorRead01 = 246,
ScsiErrorRead02 = 247,
ScsiError012 = 248,
ScsiError013 = 249,
ScsiErrorCirc01 = 250,
```

```
ScsiErrorCrc01 = 251,
ScsiErrorDecom01 = 252,
ScsiErrorDrive01 = 253,
ScsiErrorDrive02 = 254,
ScsiError014 = 255,
ScsiError015 = 256,
ScsiError016 = 257,
ScsiError017 = 258,
ScsiErrorMech01 = 259,
ScsiErrorMech02 = 260,
ScsiErrorRecover01 = 261,
ScsiErrorRecover02 = 262,
ScsiErrorRecover03 = 263,
ScsiErrorRecover04 = 264,
ScsiErrorRecover05 = 265,
ScsiErrorRecover06 = 266,
ScsiErrorRecover07 = 267,
ScsiErrorRecover08 = 268,
ScsiErrorRecover09 = 269,
ScsiErrorRecover10 = 270,
ScsiErrorRecover11 = 271,
ScsiErrorRecover12 = 272,
ScsiErrorRecover13 = 273,
ScsiErrorRecover14 = 274,
ScsiErrorRecover15 = 275,
ScsiErrorRecover16 = 276,
ScsiErrorRecover17 = 277,
ScsiError018 = 278,
ScsiError019 = 279,
ScsiError020 = 280,
ScsiError021 = 281,
ScsiError022 = 282,
ScsiError023 = 283,
ScsiError024 = 284,
ScsiErrorCdb01 = 285,
ScsiErrorCdb02 = 286,
ScsiErrorUnit16 = 287,
ScsiErrorParam01 = 288,
ScsiErrorParam02 = 289,
ScsiErrorParam03 = 290,
ScsiErrorParam04 = 291,
ScsiError025 = 292,
ScsiError026 = 293,
ScsiErrorSegm01 = 296,
ScsiErrorSegm02 = 297,
ScsiErrorSegm03 = 298,
ScsiError027 = 299,
ScsiError028 = 300,
ScsiErrorSegm04 = 301,
ScsiErrorWrite06 = 302,
ScsiErrorWrite07 = 303,
ScsiErrorWrite08 = 305,
ScsiErrorWrite09 = 306,
ScsiErrorWrite10 = 307,
ScsiErrorWrite11 = 308,
ScsiError029 = 309,
ScsiError030 = 310,
ScsiError031 = 311,
ScsiError032 = 312,
ScsiError033 = 313,
ScsiError034 = 314,
ScsiError035 = 315,
ScsiError036 = 316,
ScsiError037 = 317,
ScsiErrorParam05 = 318,
ScsiErrorParam06 = 319,
ScsiErrorParam07 = 320,
ScsiError038 = 321,
ScsiError039 = 322,
ScsiError040 = 323,
ScsiError041 = 324,
ScsiErrorCommand02 = 325,
```

```
ScsiError042 = 326,
ScsiError043 = 327,
ScsiError044 = 328,
ScsiError045 = 329,
ScsiErrorCommand03 = 330,
ScsiErrorDisk01 = 331,
ScsiErrorDisk02 = 332,
ScsiErrorDisk03 = 333,
ScsiErrorDisk04 = 334,
ScsiErrorDisk05 = 335,
ScsiErrorDisk06 = 336,
ScsiErrorDisk07 = 337,
ScsiErrorDisk08 = 338,
ScsiErrorDisk09 = 339,
ScsiErrorDisk10 = 340,
ScsiErrorDisk11 = 341,
ScsiErrorDisk12 = 342,
ScsiErrorDisk13 = 343,
ScsiError046 = 344,
ScsiError047 = 345,
ScsiError048 = 346,
ScsiError049 = 347,
ScsiError050 = 348,
ScsiError051 = 349,
ScsiError052 = 350,
ScsiErrorParam08 = 351,
ScsiErrorParam09 = 352,
ScsiErrorDisk14 = 353,
ScsiErrorDisk15 = 354,
ScsiErrorDisk16 = 355,
ScsiErrorDisk17 = 356,
ScsiErrorDisk18 = 357,
ScsiErrorDisk19 = 358,
ScsiErrorDisk20 = 359,
ScsiErrorDisk21 = 360,
ScsiErrorDisk22 = 361,
ScsiErrorDisk23 = 362,
ScsiErrorDisk24 = 363,
ScsiErrorDisk25 = 364,
ScsiErrorDisk26 = 365,
ScsiErrorMech03 = 366,
ScsiError053 = 367,
ScsiErrorUnit17 = 368,
ScsiErrorUnit18 = 369,
ScsiErrorUnit19 = 370,
ScsiErrorUnit20 = 371,
ScsiErrorUnit21 = 372,
ScsiError054 = 373,
ScsiError055 = 374,
ScsiError056 = 375,
ScsiError057 = 376,
ScsiError058 = 377,
ScsiError059 = 378,
ScsiError060 = 379,
ScsiError061 = 380,
ScsiError062 = 381,
ScsiError063 = 382,
ScsiErrorVol01 = 383,
ScsiErrorVol02 = 384,
ScsiErrorVol03 = 385,
ScsiErrorVol04 = 386,
ScsiError064 = 387,
ScsiError065 = 388,
ScsiErrorDisk27 = 389,
ScsiErrorDisk28 = 390,
ScsiError066 = 391,
ScsiError067 = 392,
ScsiError068 = 393,
ScsiError069 = 394,
ScsiError070 = 395,
ScsiError071 = 396,
ScsiError072 = 397,
```

```
ScsiError073 = 398,
ScsiError074 = 399,
ScsiError075 = 400,
ScsiError076 = 401,
ScsiErrorCommand04 = 402,
ScsiError077 = 403,
ScsiErrorUnit22 = 404,
ScsiErrorCommand05 = 405,
ScsiError078 = 410,
ScsiError079 = 411,
ScsiError080 = 412,
ScsiError081 = 413,
ScsiError082 = 414,
ScsiError083 = 415,
ScsiError084 = 416,
ScsiError085 = 417,
ScsiErrorLog02 = 418,
ScsiError086 = 419,
ScsiErrorLog03 = 420,
ScsiErrorLog04 = 421,
ScsiError087 = 422,
ScsiErrorUnit23 = 424,
ScsiError088 = 425,
ScsiError089 = 426,
ScsiError090 = 427,
ScsiError091 = 428,
ScsiError092 = 429,
ScsiError093 = 430,
ScsiError094 = 431,
ScsiError095 = 432,
ScsiError096 = 433,
ScsiError097 = 434,
ScsiError098 = 435,
ScsiError099 = 436,
ScsiErrorDcss01 = 437,
ScsiErrorDcss02 = 438,
ScsiErrorDcss03 = 439,
ScsiErrorDcss04 = 440,
ScsiErrorDcss05 = 441,
ScsiErrorDcss06 = 442,
ScsiErrorSession01 = 443,
ScsiErrorSession02 = 444,
ScsiErrorSession03 = 445,
ScsiErrorSession04 = 446,
ScsiError100 = 447,
ScsiError101 = 448,
ScsiError102 = 449,
ScsiError103 = 450,
ScsiError104 = 451,
ScsiError105 = 452,
ScsiErrorAlloc01 = 453,
ScsiErrorAlloc02 = 454,
ScsiError106 = 455,
ScsiErrorDisk34 = 456,
MultiError01 = 457,
MultiError02 = 458,
ScsiErrorDisk35 = 459,
ScsiError108 = 460,
ScsiError109 = 461,
ScsiErrorAtt04 = 462,
ScsiErrorAtt05 = 463,
ScsiError110 = 464,
ScsiError111 = 465,
ScsiError112 = 466,
ScsiErrorDisk36 = 467,
ScsiError113 = 468,
ScsiError114 = 469,
ScsiError115 = 470,
ScsiError116 = 471,
ScsiError117 = 472,
ScsiError118 = 473,
ScsiError119 = 474,
```

```
ScsiErrorDcss07 = 475,
ScsiErrorDcss08 = 476,
SdkMessageWait = 500,
SdkMessageWritestart = 501,
SdkMessageEreasestart = 502,
SdkMessageExtrFile = 503,
SdkMessageSimulate = 504,
SdkMessageImport = 505,
SdkMessageFormat = 506,
SdkMessageFormatDone = 507,
SdkMessageImagecreatestart = 508,
SdkMessage01 = 600,
SdkMessage02 = 601,
SdkMessage03 = 602,
SdkMessage04 = 603,
SdkMessage05 = 604,
SdkMessage06 = 605,
SdkMessage07 = 606,
SdkMessage08 = 607,
SdkMessage10 = 608,
SdkMessage11 = 609,
SdkMessage12 = 610,
SdkMessage13 = 611,
SdkMessage14 = 612,
SdkMessage15 = 613,
SdkMessage16 = 614,
GuiResource01 = 630,
GuiResource02 = 631,
GuiResource03 = 632,
GuiResource04 = 633,
GuiResource05 = 634,
GuiResource06 = 635,
GuiResource07 = 636,
GuiResource08 = 637,
GuiResource09 = 638,
GuiResource10 = 639,
GuiResource11 = 640,
GuiResource12 = 641,
GuiResource13 = 642,
GuiResource14 = 643,
GuiResource15 = 644,
GuiResource16 = 645,
GuiResource17 = 646,
GuiResource18 = 647,
GuiResource19 = 648,
GuiResource20 = 649,
GuiResource21 = 650,
GuiResource22 = 651,
GuiResource23 = 652,
GuiResource24 = 653,
GuiResource25 = 654,
GuiResource26 = 655,
GuiResource27 = 656,
GuiResource28 = 657,
GuiResource29 = 658,
GuiResource30 = 659,
GuiResource31 = 660,
GuiResource32 = 661,
GuiResource33 = 662,
GuiResource34 = 663,
GuiResource35 = 664,
GuiResource36 = 665,
GuiResource37 = 666,
GuiResource38 = 667,
GuiResource39 = 668,
GuiResource40 = 669,
GuiResource41 = 670,
GuiResource42 = 671,
GuiResource43 = 672,
GuiResource44 = 673,
GuiResource45 = 674,
GuiResource46 = 675,
```

```
    GuiError01 = 676,
    GuiError02 = 677,
    GuiResource47 = 678,
    GuiResource48 = 679,
    GuiResource49 = 680,
    GuiResource50 = 681,
    SdkVerifyErrorHddfileunreadable = 700,
    SdkVerifyErrorCdfileunreadable = 701,
    SdkVerifyErrorFilesdifferent = 702
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
SdkErrorNo = 0	IsoSDK (see page 1) Error code: No error occurred
SdkErrorNotAllowed = 1	IsoSDK (see page 1) Error code: Command not allowed. Check the API guide for possible miss configuration.
ScsiErrorParam10 = 2	IsoSDK (see page 1) Error code: A bad parameter has been passed in
SdkErrorBadRequest = 3	IsoSDK (see page 1) Error code: Bad request
SdkErrorGeneral = 4	IsoSDK (see page 1) Error code: A general error occurred
ScsiErrorAspi01 = 5	IsoSDK (see page 1) Error code: An ASPI error occurred
ScsiErrorAspi06 = 6	IsoSDK (see page 1) Error code: Invalid device ID or device is not a CD or DVD drive
SdkErrorInvalidPath = 7	IsoSDK (see page 1) Error code: An invalid path has been passed in.
SdkErrorInvalidSrcPath = 8	IsoSDK (see page 1) Error code: An invalid source path has been passed in.
SdkErrorInvalidDestPath = 9	IsoSDK (see page 1) Error code: An invalid destination path has been passed in.
SdkErrorInvalidFileName = 10	IsoSDK (see page 1) Error code: An invalid file name has been passed in.
SdkErrorPathExists = 11	IsoSDK (see page 1) Error code: The path already exists.
SdkErrorFileExists = 12	IsoSDK (see page 1) Error code: The file already exists
SdkCueErrorCommand06 = 13	IsoSDK (see page 1) Error code: Command failed.
SdkErrorInvalidFileFormat = 15	IsoSDK (see page 1) Error code: The file has an invalid format.
SdkErrorFileOpen = 16	IsoSDK (see page 1) Error code: File could not be opened.
SdkErrorCorruptOrInvalidCueFile = 17	IsoSDK (see page 1) Error code: The cue file is corrupt or completely invalid.
SdkErrorBinFileNotFound = 18	IsoSDK (see page 1) Error code: The bin file could not be found.
SdkErrorNotImplemented = 19	IsoSDK (see page 1) Error code: Not implemented.
SdkErrorNotAllowedForThisBurner = 20	IsoSDK (see page 1) Error code: Not allowed for the current burner.
ScsiErrorDisk29 = 21	IsoSDK (see page 1) Error code: The disk is not writable.
ScsiErrorDisk30 = 22	IsoSDK (see page 1) Error code: Wrong medium.
ScsiErrorDisk31 = 23	IsoSDK (see page 1) Error code: No medium.
ScsiErrorDisk32 = 24	IsoSDK (see page 1) Error code: The disk is not erasable.
SdkErrorUnknownTextid = 25	IsoSDK (see page 1) Error code: The text ID that has been passed in is unknown.
SdkErrorMoreSpaceNeeded = 26	IsoSDK (see page 1) Error code: The supplied buffer is too small.

SdkErrorBurnInProgress = 27	IsoSDK (see page 1) error code: Not allowed while burning.
SdkErrorInvalidSessionNumber = 28	IsoSDK (see page 1) Error code: Invalid session number has been passed in.
SdkErrorEmptyPassword = 29	IsoSDK (see page 1) Error code: Empty passwords are not allowed.
SdkErrorUnknown = 101	IsoSDK (see page 1) Error code: An unknown error occurred.
ScsiErrorAspi02 = 102	IsoSDK (see page 1) Error code: Invalid host adapter.
ScsiErrorAspi03 = 103	IsoSDK (see page 1) Error code: No device has been specified.
SdkErrorInvalidsrb = 104	IsoSDK (see page 1) Error code: Invalid SRB
SdkErrorBufferalignment = 105	IsoSDK (see page 1) Error code: Buffer alignment.
ScsiErrorAspi04 = 106	IsoSDK (see page 1) Error code: ASPI is busy.
SdkErrorBuffertoobig = 107	IsoSDK (see page 1) Error code: The buffer is too big.
SdkErrorTimeout = 108	IsoSDK (see page 1) Error code: The buffer is too big.
SdkErrorSrbtimeout = 109	IsoSDK (see page 1) Error code: A SRB timeout occurred.
SdkErrorMessagereject = 110	IsoSDK (see page 1) Error code: Message has been rejected.
ScsiErrorAspi05 = 111	IsoSDK (see page 1) Error code: Bus reset.
SdkErrorParityerr = 112	IsoSDK (see page 1) Error code: Parity error.
ScsiError107 = 113	IsoSDK (see page 1) Error code: Failed to request sense.
SdkErrorSelectiontimeout = 114	IsoSDK (see page 1) Error code: Selection timed out.
SdkErrorDataoverrun = 115	IsoSDK (see page 1) Error code: A data-overrun occurred.
SdkErrorUnexpectedbusfree = 116	IsoSDK (see page 1) Error code: Unexpected bus free.
SdkErrorCheckcondition = 117	IsoSDK (see page 1) Error code: Check condition.
ScsiErrorAspi08 = 118	IsoSDK (see page 1) Error code: The target is busy.
ScsiErrorAspi09 = 119	IsoSDK (see page 1) Error code: Target reservation conflict.
ScsiErrorAspi10 = 120	IsoSDK (see page 1) Error code: The target queue is full.
SdkErrorDatarecovererror = 121	IsoSDK (see page 1) Error code: Unable to recover data.
SdkErrorNotready = 122	IsoSDK (see page 1) Error code: Not ready.
ScsiErrorDisk33 = 123	IsoSDK (see page 1) Error code: Medium error.
SdkErrorHardwareerror = 124	IsoSDK (see page 1) Error code: Hardware error.
SdkErrorUnitattention = 126	IsoSDK (see page 1) Error code: Unit attention.
SdkErrorDataprotect = 127	IsoSDK (see page 1) Error code: Data protect.
SdkErrorErasecheck = 128	IsoSDK (see page 1) Error code: Erase check.
SdkErrorCopyaborted = 129	IsoSDK (see page 1) Error code: Copy aborted.
SdkErrorAbortedcommand = 130	IsoSDK (see page 1) Error code: Aborted command.
SdkErrorVolumeoverflow = 131	IsoSDK (see page 1) Error code: Volume overflow.
SdkErrorMiscompare = 132	IsoSDK (see page 1) Error code: Miscompare.
SdkErrorReserved = 133	IsoSDK (see page 1) Error code: Reserved.
SdkErrorFilemark = 134	IsoSDK (see page 1) Error code: Filemark error.
SdkErrorIllegallength = 136	IsoSDK (see page 1) Error code: Illegal length.
SdkErrorIncorrectlength = 137	IsoSDK (see page 1) Error code: Incorrect length.
SdkErrorAborted = 138	IsoSDK (see page 1) Error code: Operation aborted by user.
SdkErrorFileinuse = 139	IsoSDK (see page 1) Error code: Could not open file.
SdkErrorCreatefile = 140	IsoSDK (see page 1) Error code: Couldn't create file. File in use or exists or nor user rights.
ScsiErrorAspi07 = 141	IsoSDK (see page 1) Error code: The device is busy.

ScsiErrorWrite12 = 142	IsoSDK (see page 1) Error code: A write-error occurred.
SdkErrorNotsupported = 143	IsoSDK (see page 1) Error code: Not supported or not longer available.
SdkErrorNextaddress = 144	IsoSDK (see page 1) Error code: Error getting next writable address.
SdkErrorTooMuchData = 145	IsoSDK (see page 1) Error code: Too much data for this mode.
SdkErrorMaxdirs = 146	IsoSDK (see page 1) Error code: Not more directories are allowed.
SdkErrorMaxfiles = 147	IsoSDK (see page 1) Error code: Not more files are allowed.
SdkErrorErrinvalidfilename = 148	IsoSDK (see page 1) Error code: Error importing session, file name length exceeds 120 chars.
SdkErrorImportsession = 149	IsoSDK (see page 1) Error code: Error importing session.
SdkErrorIsoimagenotfound = 150	IsoSDK (see page 1) Error code: The selected ISO Image was not found.
SdkIntError1 = 151	IsoSDK (see page 1) Error code: Internal error 1.
SdkIntError2 = 152	IsoSDK (see page 1) Error code: Internal error 2.
SdkIntError3 = 153	IsoSDK (see page 1) Error code: Internal error 3.
SdkIntError4 = 154	IsoSDK (see page 1) Error code: Internal error 4.
SdkIntError5 = 155	IsoSDK (see page 1) Error code: Internal error 5.
SdkCueErrorSendingCue = 156	IsoSDK (see page 1) Error code: Send cue sheet failed.
SdkCueErrorUeol = 157	IsoSDK (see page 1) Error code: Unexpected end of line.
SdkCueErrorField = 158	IsoSDK (see page 1) Error code: Invalid field
SdkCueErrorFile = 159	IsoSDK (see page 1) Error code: File not found.
SdkCueErrorCommand01 = 160	IsoSDK (see page 1) Error code: Invalid command.
SdkIntErrorFormat = 161	IsoSDK (see page 1) Error code: Error formatting DVD +/- RW medium.
SdkErrorNotAllowedForThisUdfVersion = 162	IsoSDK (see page 1) Error code: Invalid option for this UDF version.
SdkErrorInvalidUdfVersion = 163	IsoSDK (see page 1) Error code: Invalid UDF version number.
SdkErrorMp3libNotFound = 164	IsoSDK (see page 1) Error code: mp3lib/basslib not found.
SdkErrorIncompatibleFsType = 165	IsoSDK (see page 1) Error code: Selected project type is incompatible with last recorded session.
SdkErrorCompressionConflict = 166	IsoSDK (see page 1) Error code: Compression conflict.
SdkErrorEncryptionConflict = 167	IsoSDK (see page 1) Error code: Encryption conflict.
SdkErrorInvalidDirIndex = 168	IosSDK Error code: Invalid directory index.
SdkErrorInvalidMcn = 169	IsoSDK (see page 1) Error code: The MCN Code inside the TrackInformation is invalid.
SdkErrorInvalidIsrc = 170	IsoSDK (see page 1) Error code: The SRC Code inside the TrackInformation is invalid.
IleTooLongDirectoryNesting = 171	ISO Level Error: To long Directory.
IleTooBigFile = 172	ISO Level Error: File is to Big.
SdkErrorInvalidIndex = 173	IsoSDK (see page 1) Error code: The sub index of the audio track is wrong.
SdkErrorTooMuchIndexes = 174	IsoSDK (see page 1) Error code: The amount of indexes inside a AudioFileProperty (see page 141) is to much. Max. 99
SdkErrorEnclibNotFound = 175	IsoSDK (see page 1) Error code: The enclib.dll for tagging and encoding operations was not found.
SdkErrorCdTextNotFound = 176	IsoSDK (see page 1) Error code: No CD-Text info found on the disk.

SdkErrorNetTagsDisk = 177	IsoSDK (see page 1) Error code: Compact disc error during forming request to internet tags' DB.
SdkErrorNetTagsConnect = 178	IsoSDK (see page 1) Error code: Connection error to internet tags' DB, check internet connection and settings.
SdkErrorNetTagsServer = 179	IsoSDK (see page 1) Error code: Internet tags' DB server error.
SdkErrorNetTagsNoMatch = 180	IsoSDK (see page 1) Error code: No match found in internet tags' DB.
SdkErrorNetTagsInternal = 181	IsoSDK (see page 1) Error code: Internal error during internet tags' DB processing.
ScsiError01 = 200	IsoSDK (see page 1) Error code: No additional sense information.
ScsiError02 = 201	IsoSDK (see page 1) Error code: I/O process terminated.
ScsiErrorAudio01 = 202	IsoSDK (see page 1) Error code: Audio play operation in progress.
ScsiErrorAudio02 = 203	IsoSDK (see page 1) Error code: Audio play operation paused.
ScsiErrorAudio03 = 204	IsoSDK (see page 1) Error code: Audio play operation successfully completed.
ScsiErrorAudio04 = 205	IsoSDK (see page 1) Error code: Audio play operation stopped due to error.
ScsiErrorAudio05 = 206	IsoSDK (see page 1) Error code: No current audio status to return.
ScsiError001 = 207	IsoSDK (see page 1) Error code: Operation in progress.
ScsiError002 = 208	IsoSDK (see page 1) Error code: Cleaning requested.
ScsiErrorUnit01 = 209	IsoSDK (see page 1) Error code: No seek complete.
ScsiErrorUnit02 = 210	IsoSDK (see page 1) Error code: Logical unit not ready, cause not reportable.
ScsiErrorUnit03 = 211	IsoSDK (see page 1) Error code: Logical unit is in process of becoming ready.
ScsiErrorUnit04 = 212	IsoSDK (see page 1) Error code: Logical unit not ready, initializing command required.
ScsiErrorUnit05 = 213	IsoSDK (see page 1) Error code: Logical unit not ready, manual intervention required.
ScsiErrorUnit06 = 214	IsoSDK (see page 1) Error code: Logical unit not ready, format in progress.
ScsiErrorUnit07 = 215	IsoSDK (see page 1) Error code: Logical unit not ready, operation in progress.
ScsiErrorUnit08 = 216	IsoSDK (see page 1) Error code: Logical unit not ready, long write in progress.
ScsiErrorUnit09 = 217	IsoSDK (see page 1) Error code: Logical unit not ready, self-test in progress.
ScsiErrorUnit10 = 218	IsoSDK (see page 1) Error code: Logical unit does not respond to selection.
ScsiError004 = 219	IsoSDK (see page 1) Error code: No reference position found.
ScsiError005 = 220	IsoSDK (see page 1) Error code: Multiple peripheral devices selected.
ScsiErrorUnit11 = 221	IsoSDK (see page 1) Error code: Logical unit communication failure.
ScsiErrorUnit12 = 222	IsoSDK (see page 1) Error code: Logical unit communication time-out.
ScsiErrorUnit13 = 223	IsoSDK (see page 1) Error code: Logical unit communication parity error.

ScsiErrorUnit14 = 224	IsoSDK (see page 1) Error code: Logical unit communication crc error (ultra-dma/32).
ScsiError006 = 225	IsoSDK (see page 1) Error code: Unreachable copy target.
ScsiError007 = 226	IsoSDK (see page 1) Error code: Track following error.
ScsiError008 = 227	IsoSDK (see page 1) Error code: Tracking servo failure.
ScsiError009 = 228	IsoSDK (see page 1) Error code: Focus servo failure.
ScsiError010 = 229	IsoSDK (see page 1) Error code: Spindle servo failure.
ScsiError011 = 230	IsoSDK (see page 1) Error code: Head select fault.
ScsiErrorLog01 = 231	IsoSDK (see page 1) Error code: Error log overflow.
ScsiErrorAtt01 = 232	IsoSDK (see page 1) Error code: Warning.
ScsiErrorAtt02 = 233	IsoSDK (see page 1) Error code: Warning - Specified temperature exceeded.
ScsiErrorAtt03 = 234	IsoSDK (see page 1) Error code: Warning - Enclosure degraded.
ScsiErrorWrite01 = 235	IsoSDK (see page 1) Error code: Write error.
ScsiErrorWrite02 = 236	IsoSDK (see page 1) Error code: Write error - Recovery needed.
ScsiErrorWrite03 = 237	IsoSDK (see page 1) Error code: Write error - Recovery failed.
ScsiErrorWrite04 = 238	IsoSDK (see page 1) Error code: Write error - Loss of streaming.
ScsiErrorWrite05 = 239	IsoSDK (see page 1) Error code: Write error - Padding blocks added.
ScsiErrorExt01 = 240	IsoSDK (see page 1) Error code: Error detected by third party temporary initiator.
ScsiErrorExt02 = 241	IsoSDK (see page 1) Error code: Third party device failure.
ScsiErrorTarget01 = 242	IsoSDK (see page 1) Error code: Copy target device not reachable.
ScsiErrorTarget02 = 243	IsoSDK (see page 1) Error code: Incorrect copy target device type.
ScsiErrorTarget03 = 244	IsoSDK (see page 1) Error code: Copy target device data underrun.
ScsiErrorTarget04 = 245	IsoSDK (see page 1) Error code: Copy target device data overrun.
ScsiErrorRead01 = 246	IsoSDK (see page 1) Error code: Unrecovered read error.
ScsiErrorRead02 = 247	IsoSDK (see page 1) Error code: Read retries exhausted.
ScsiError012 = 248	IsoSDK (see page 1) Error code: Error too long to correct.
ScsiError013 = 249	IsoSDK (see page 1) Error code: I-ec uncorrectable error.
ScsiErrorCirc01 = 250	IsoSDK (see page 1) Error code: Circ unrecovered error.
ScsiErrorCrc01 = 251	IsoSDK (see page 1) Error code: De-compression crc error.
ScsiErrorDecom01 = 252	IsoSDK (see page 1) Error code: Cannot decompress using declared algorithm.
ScsiErrorDrive01 = 253	IsoSDK (see page 1) Error code: Error reading UPC/EAN number.
ScsiErrorDrive02 = 254	IsoSDK (see page 1) Error code: Error reading ISRC number.
ScsiError014 = 255	IsoSDK (see page 1) Error code: Read error - Loss of streaming.
ScsiError015 = 256	IsoSDK (see page 1) Error code: Recorded entity not found.
ScsiError016 = 257	IsoSDK (see page 1) Error code: Record not found.
ScsiError017 = 258	IsoSDK (see page 1) Error code: Random positioning error.

ScsiErrorMech01 = 259	IsoSDK (see page 1) Error code: Mechanical positioning error.
ScsiErrorMech02 = 260	IsoSDK (see page 1) Error code: Positioning error detected by read of medium.
ScsiErrorRecover01 = 261	IsoSDK (see page 1) Error code: Recovered data with no error correction applied.
ScsiErrorRecover02 = 262	IsoSDK (see page 1) Error code: Recovered data with retries.
ScsiErrorRecover03 = 263	IsoSDK (see page 1) Error code: Recovered data with positive head offset.
ScsiErrorRecover04 = 264	IsoSDK (see page 1) Error code: Recovered data with negative head offset.
ScsiErrorRecover05 = 265	IsoSDK (see page 1) Error code: Recovered data with retries and/or circ applied.
ScsiErrorRecover06 = 266	IsoSDK (see page 1) Error code: Recovered data using previous sector id.
ScsiErrorRecover07 = 267	IsoSDK (see page 1) Error code: Recovered data without ecc - recommend reassignment.
ScsiErrorRecover08 = 268	IsoSDK (see page 1) Error code: Recovered data without ecc - recommend rewrite.
ScsiErrorRecover09 = 269	IsoSDK (see page 1) Error code: Recovered data without ecc - data rewritten.
ScsiErrorRecover10 = 270	IsoSDK (see page 1) Error code: Recovered data with error correction applied.
ScsiErrorRecover11 = 271	IsoSDK (see page 1) Error code: Recovered data with error corr. & retries applied. s
ScsiErrorRecover12 = 272	IsoSDK (see page 1) Error code: Recovered data - data auto-reallocated.
ScsiErrorRecover13 = 273	IsoSDK (see page 1) Error code: Recovered data with circ.
ScsiErrorRecover14 = 274	IsoSDK (see page 1) Error code: Recovered data with l-ec.
ScsiErrorRecover15 = 275	IsoSDK (see page 1) Error code: Recovered data - recommend reassignment.
ScsiErrorRecover16 = 276	IsoSDK (see page 1) Error code: Recovered data - recommend rewrite.
ScsiErrorRecover17 = 277	IsoSDK (see page 1) Error code: Recovered data with linking.
ScsiError018 = 278	IsoSDK (see page 1) Error code: Parameter list length error.
ScsiError019 = 279	IsoSDK (see page 1) Error code: Synchronous data transfer error.
ScsiError020 = 280	IsoSDK (see page 1) Error code: Miscompare during verify operation.
ScsiError021 = 281	IsoSDK (see page 1) Error code: Invalid command operation code.
ScsiError022 = 282	IsoSDK (see page 1) Error code: Logical block address out of range.
ScsiError023 = 283	IsoSDK (see page 1) Error code: Invalid element address.
ScsiError024 = 284	IsoSDK (see page 1) Error code: Invalid address for write.
ScsiErrorCdb01 = 285	IsoSDK (see page 1) Error code: Invalid field in CDB.
ScsiErrorCdb02 = 286	IsoSDK (see page 1) Error code: CDB decryption error.
ScsiErrorUnit16 = 287	IsoSDK (see page 1) Error code: Logical unit not supported.
ScsiErrorParam01 = 288	IsoSDK (see page 1) Error code: Invalid field in parameter list.
ScsiErrorParam02 = 289	IsoSDK (see page 1) Error code: Parameter not supported.

ScsiErrorParam03 = 290	IsoSDK (see page 1) Error code: Parameter value invalid.
ScsiErrorParam04 = 291	IsoSDK (see page 1) Error code: Threshold parameters not supported.
ScsiError025 = 292	IsoSDK (see page 1) Error code: Invalid release of persistent reservation.
ScsiError026 = 293	IsoSDK (see page 1) Error code: Data decryption error.
ScsiErrorSegm01 = 296	IsoSDK (see page 1) Error code: Too many segment descriptors.
ScsiErrorSegm02 = 297	IsoSDK (see page 1) Error code: Unsupported segment descriptor type code.
ScsiErrorSegm03 = 298	IsoSDK (see page 1) Error code: Unexpected inexact segment.
ScsiError027 = 299	IsoSDK (see page 1) Error code: Inline data length exceeded.
ScsiError028 = 300	IsoSDK (see page 1) Error code: Invalid operation for copy source or destination.
ScsiErrorSegm04 = 301	IsoSDK (see page 1) Error code: Copy segment granularity violation.
ScsiErrorWrite06 = 302	IsoSDK (see page 1) Error code: Write protected.
ScsiErrorWrite07 = 303	IsoSDK (see page 1) Error code: Hardware write protected.
ScsiErrorWrite08 = 305	IsoSDK (see page 1) Error code: Associated write protect.
ScsiErrorWrite09 = 306	IsoSDK (see page 1) Error code: Persistent write protect.
ScsiErrorWrite10 = 307	IsoSDK (see page 1) Error code: Permanent write protect.
ScsiErrorWrite11 = 308	IsoSDK (see page 1) Error code: Conditional write protect.
ScsiError029 = 309	IsoSDK (see page 1) Error code: Not ready to ready change, medium may have changed.
ScsiError030 = 310	IsoSDK (see page 1) Error code: Import or export element accessed.
ScsiError031 = 311	IsoSDK (see page 1) Error code: Power on, reset, or bus device reset occurred.
ScsiError032 = 312	IsoSDK (see page 1) Error code: Power on occurred.
ScsiError033 = 313	IsoSDK (see page 1) Error code: SCSI bus reset occurred.
ScsiError034 = 314	IsoSDK (see page 1) Error code: Bus device reset function occurred.
ScsiError035 = 315	IsoSDK (see page 1) Error code: Device internal reset.
ScsiError036 = 316	IsoSDK (see page 1) Error code: Transceiver mode changed to single-ended.
ScsiError037 = 317	IsoSDK (see page 1) Error code: Transceiver mode changed to LVD.
ScsiErrorParam05 = 318	IsoSDK (see page 1) Error code: Parameters changed.
ScsiErrorParam06 = 319	IsoSDK (see page 1) Error code: Parameters changed.
ScsiErrorParam07 = 320	IsoSDK (see page 1) Error code: Log parameters changed. s
ScsiError038 = 321	IsoSDK (see page 1) Error code: Reservations pre-empted .
ScsiError039 = 322	IsoSDK (see page 1) Error code: Reservations released.
ScsiError040 = 323	IsoSDK (see page 1) Error code: Registrations pre-empted.
ScsiError041 = 324	IsoSDK (see page 1) Error code: Copy cannot execute since host cannot disconnect.
ScsiErrorCommand02 = 325	IsoSDK (see page 1) Error code: Command sequence error.
ScsiError042 = 326	IsoSDK (see page 1) Error code: Current program area is not empty.

ScsiError043 = 327	IsoSDK (see page 1) Error code: Current program area is empty.
ScsiError044 = 328	IsoSDK (see page 1) Error code: Persistent prevent conflict.
ScsiError045 = 329	IsoSDK (see page 1) Error code: Insufficient time for operation.
ScsiErrorCommand03 = 330	IsoSDK (see page 1) Error code: Commands cleared by another initiator.
ScsiErrorDisk01 = 331	IsoSDK (see page 1) Error code: Incompatible medium installed.
ScsiErrorDisk02 = 332	IsoSDK (see page 1) Error code: Cannot read medium - unknown format.
ScsiErrorDisk03 = 333	IsoSDK (see page 1) Error code: Cannot read medium - incompatible format.
ScsiErrorDisk04 = 334	IsoSDK (see page 1) Error code: Cleaning cartridge installed.
ScsiErrorDisk05 = 335	IsoSDK (see page 1) Error code: Cannot write medium - unknown format.
ScsiErrorDisk06 = 336	IsoSDK (see page 1) Error code: Cannot write medium - incompatible format.
ScsiErrorDisk07 = 337	IsoSDK (see page 1) Error code: Cannot format medium - incompatible medium.
ScsiErrorDisk08 = 338	IsoSDK (see page 1) Error code: Cleaning failure.
ScsiErrorDisk09 = 339	IsoSDK (see page 1) Error code: Cannot write - application code mismatch.
ScsiErrorDisk10 = 340	IsoSDK (see page 1) Error code: Current session not fixated for append.
ScsiErrorDisk11 = 341	IsoSDK (see page 1) Error code: Medium not formatted.
ScsiErrorDisk12 = 342	IsoSDK (see page 1) Error code: Medium format corrupted.
ScsiErrorDisk13 = 343	IsoSDK (see page 1) Error code: Format command failed.
ScsiError046 = 344	IsoSDK (see page 1) Error code: Zoned formatting failed due to spare linking.
ScsiError047 = 345	IsoSDK (see page 1) Error code: Enclosure failure.
ScsiError048 = 346	IsoSDK (see page 1) Error code: Enclosure services failure.
ScsiError049 = 347	IsoSDK (see page 1) Error code: Unsupported enclosure function.
ScsiError050 = 348	IsoSDK (see page 1) Error code: Enclosure services unavailable.
ScsiError051 = 349	IsoSDK (see page 1) Error code: Enclosure services transfer failure.
ScsiError052 = 350	IsoSDK (see page 1) Error code: Enclosure services transfer refused.
ScsiErrorParam08 = 351	IsoSDK (see page 1) Error code: Rounded parameter.
ScsiErrorParam09 = 352	IsoSDK (see page 1) Error code: Saving parameters not supported.
ScsiErrorDisk14 = 353	IsoSDK (see page 1) Error code: Medium not present.
ScsiErrorDisk15 = 354	IsoSDK (see page 1) Error code: Medium not present - tray closed.
ScsiErrorDisk16 = 355	IsoSDK (see page 1) Error code: Medium not present - tray open.
ScsiErrorDisk17 = 356	IsoSDK (see page 1) Error code: Medium not present - loadable.
ScsiErrorDisk18 = 357	IsoSDK (see page 1) Error code: Medium not present - medium auxiliary memory accessible.

ScsiErrorDisk19 = 358	IsoSDK (see page 1) Error code: Medium destination element full.
ScsiErrorDisk20 = 359	IsoSDK (see page 1) Error code: Medium source element empty.
ScsiErrorDisk21 = 360	IsoSDK (see page 1) Error code: End of medium reached.
ScsiErrorDisk22 = 361	IsoSDK (see page 1) Error code: Medium magazine not accessible.
ScsiErrorDisk23 = 362	IsoSDK (see page 1) Error code: Medium magazine removed.
ScsiErrorDisk24 = 363	IsoSDK (see page 1) Error code: Medium magazine inserted.
ScsiErrorDisk25 = 364	IsoSDK (see page 1) Error code: Medium magazine locked.
ScsiErrorDisk26 = 365	IsoSDK (see page 1) Error code: Medium magazine unlocked.
ScsiErrorMech03 = 366	IsoSDK (see page 1) Error code: Mechanical positioning or changer error.
ScsiError053 = 367	IsoSDK (see page 1) Error code: Invalid bits in identify message.
ScsiErrorUnit17 = 368	IsoSDK (see page 1) Error code: Logical unit has not self-configured yet.
ScsiErrorUnit18 = 369	IsoSDK (see page 1) Error code: Logical unit failure.
ScsiErrorUnit19 = 370	IsoSDK (see page 1) Error code: Timeout on logical unit.
ScsiErrorUnit20 = 371	IsoSDK (see page 1) Error code: Logical unit failed self-test.
ScsiErrorUnit21 = 372	IsoSDK (see page 1) Error code: Logical unit unable to update self-test log.
ScsiError054 = 373	IsoSDK (see page 1) Error code: Target operating conditions have changed.
ScsiError055 = 374	IsoSDK (see page 1) Error code: Microcode has been changed.
ScsiError056 = 375	IsoSDK (see page 1) Error code: Changed operating definition.
ScsiError057 = 376	IsoSDK (see page 1) Error code: Inquiry data has changed.
ScsiError058 = 377	IsoSDK (see page 1) Error code: Component device attached.
ScsiError059 = 378	IsoSDK (see page 1) Error code: Device identifier changed.
ScsiError060 = 379	IsoSDK (see page 1) Error code: Redundancy group created or modified.
ScsiError061 = 380	IsoSDK (see page 1) Error code: Redundancy group deleted.
ScsiError062 = 381	IsoSDK (see page 1) Error code: Spare created or modified.
ScsiError063 = 382	IsoSDK (see page 1) Error code: Spare deleted.
ScsiErrorVol01 = 383	IsoSDK (see page 1) Error code: Volume set created or modified.
ScsiErrorVol02 = 384	IsoSDK (see page 1) Error code: Volume set deleted.
ScsiErrorVol03 = 385	IsoSDK (see page 1) Error code: Volume set de-assigned.
ScsiErrorVol04 = 386	IsoSDK (see page 1) Error code: Volume set reassigned.
ScsiError064 = 387	IsoSDK (see page 1) Error code: Reported luns data has changed.
ScsiError065 = 388	IsoSDK (see page 1) Error code: Echo buffer overwritten.
ScsiErrorDisk27 = 389	IsoSDK (see page 1) Error code: Medium loadable.

ScsiErrorDisk28 = 390	IsoSDK (see page 1) Error code: Medium auxiliary memory accessible.
ScsiError066 = 391	IsoSDK (see page 1) Error code: Message error.
ScsiError067 = 392	IsoSDK (see page 1) Error code: Internal target failure.
ScsiError068 = 393	IsoSDK (see page 1) Error code: Select or reselect failure.
ScsiError069 = 394	IsoSDK (see page 1) Error code: Unsuccessful soft reset.
ScsiError070 = 395	IsoSDK (see page 1) Error code: SCSI parity error.
ScsiError071 = 396	IsoSDK (see page 1) Error code: Data phase crc error detected.
ScsiError072 = 397	IsoSDK (see page 1) Error code: SCSI parity error detected during st data phase.
ScsiError073 = 398	IsoSDK (see page 1) Error code: Information unit crc error detected.
ScsiError074 = 399	IsoSDK (see page 1) Error code: Asynchronous information protection error detected.
ScsiError075 = 400	IsoSDK (see page 1) Error code: Initiator detected error message received.
ScsiError076 = 401	IsoSDK (see page 1) Error code: Invalid message error.
ScsiErrorCommand04 = 402	IsoSDK (see page 1) Error code: Command phase error.
ScsiError077 = 403	IsoSDK (see page 1) Error code: Data phase error.
ScsiErrorUnit22 = 404	IsoSDK (see page 1) Error code: Logical unit failed self-configuration.
ScsiErrorCommand05 = 405	IsoSDK (see page 1) Error code: Overlapped commands attempted.
ScsiError078 = 410	IsoSDK (see page 1) Error code: Insufficient reservation resources.
ScsiError079 = 411	IsoSDK (see page 1) Error code: Insufficient resources.
ScsiError080 = 412	IsoSDK (see page 1) Error code: Insufficient registration resources.
ScsiError081 = 413	IsoSDK (see page 1) Error code: Unable to recover table-of-contents.
ScsiError082 = 414	IsoSDK (see page 1) Error code: Operator request or state change input.
ScsiError083 = 415	IsoSDK (see page 1) Error code: Operator medium removal request.
ScsiError084 = 416	IsoSDK (see page 1) Error code: Operator selected write protect.
ScsiError085 = 417	IsoSDK (see page 1) Error code: Operator selected write permit.
ScsiErrorLog02 = 418	IsoSDK (see page 1) Error code: Log exception.
ScsiError086 = 419	IsoSDK (see page 1) Error code: Threshold condition met.
ScsiErrorLog03 = 420	IsoSDK (see page 1) Error code: Log counter at maximum.
ScsiErrorLog04 = 421	IsoSDK (see page 1) Error code: Log list codes exhausted.
ScsiError087 = 422	IsoSDK (see page 1) Error code: Failure prediction threshold exceeded.
ScsiErrorUnit23 = 424	IsoSDK (see page 1) Error code: Logical unit failure prediction threshold exceeded.
ScsiError088 = 425	IsoSDK (see page 1) Error code: Spare area exhaustion prediction threshold exceeded.
ScsiError089 = 426	IsoSDK (see page 1) Error code: Failure prediction threshold exceeded (false).
ScsiError090 = 427	IsoSDK (see page 1) Error code: Low power condition on.
ScsiError091 = 428	IsoSDK (see page 1) Error code: Idle condition activated by timer.

ScsiError092 = 429	IsoSDK (see page 1) Error code: Standby condition activated by timer.
ScsiError093 = 430	IsoSDK (see page 1) Error code: Idle condition activated by command.
ScsiError094 = 431	IsoSDK (see page 1) Error code: Standby condition activated by command.
ScsiError095 = 432	IsoSDK (see page 1) Error code: End of user area encountered on this track.
ScsiError096 = 433	IsoSDK (see page 1) Error code: Packet does not fit in available space.
ScsiError097 = 434	IsoSDK (see page 1) Error code: llegal mode for this track.
ScsiError098 = 435	IsoSDK (see page 1) Error code: Invalid packet size.
ScsiError099 = 436	IsoSDK (see page 1) Error code: Voltage fault.
ScsiErrorDcss01 = 437	IsoSDK (see page 1) Error code: Copy protection key exchange failure - authentication failure.
ScsiErrorDcss02 = 438	IsoSDK (see page 1) Error code: Copy protection key exchange failure - key not present.
ScsiErrorDcss03 = 439	IsoSDK (see page 1) Error code: Copy protection key exchange failure - key not established.
ScsiErrorDcss04 = 440	IsoSDK (see page 1) Error code: Read of scrambled sector without authentication.
ScsiErrorDcss05 = 441	IsoSDK (see page 1) Error code: Medium region code is mismatched to logical unit region.
ScsiErrorDcss06 = 442	IsoSDK (see page 1) Error code: Drive region must be permanent/region reset count error.
ScsiErrorSession01 = 443	IsoSDK (see page 1) Error code: Session fixation error.
ScsiErrorSession02 = 444	IsoSDK (see page 1) Error code: Session fixation error writing lead-in.
ScsiErrorSession03 = 445	IsoSDK (see page 1) Error code: Session fixation error writing lead-out.
ScsiErrorSession04 = 446	IsoSDK (see page 1) Error code: Session fixation error - incomplete track in session.
ScsiError100 = 447	IsoSDK (see page 1) Error code: Empty or partially written reserved track.
ScsiError101 = 448	IsoSDK (see page 1) Error code: No more track reservations allowed.
ScsiError102 = 449	IsoSDK (see page 1) Error code: CD control error.
ScsiError103 = 450	IsoSDK (see page 1) Error code: Power calibration area almost full.
ScsiError104 = 451	IsoSDK (see page 1) Error code: Power calibration area is full.
ScsiError105 = 452	IsoSDK (see page 1) Error code: Power calibration area error.
ScsiErrorAlloc01 = 453	IsoSDK (see page 1) Error code: Program memory area update failure.
ScsiErrorAlloc02 = 454	IsoSDK (see page 1) Error code: Program memory area update failure.
ScsiError106 = 455	IsoSDK (see page 1) Error code: RMA/PMA is almost full.
ScsiErrorDisk34 = 456	IsoSDK (see page 1) Error code: Erase disk before writing.
MultiError01 = 457	IsoSDK (see page 1) Error code: The disks are not identical.
MultiError02 = 458	IsoSDK (see page 1) Error code: Invalid write speed.
ScsiErrorDisk35 = 459	IsoSDK (see page 1) Error code: The disk is not writable with current write method.

ScsiError108 = 460	IsoSDK (see page 1) Error code: Medium load or eject failed.
ScsiError109 = 461	IsoSDK (see page 1) Error code: Medium removal prevented.
ScsiErrorAtt04 = 462	IsoSDK (see page 1) Error code: Warning - background self-test failed.
ScsiErrorAtt05 = 463	IsoSDK (see page 1) Error code: Warning – background pre-scan detected medium error.
ScsiError110 = 464	IsoSDK (see page 1) Error code: Invalid write crossing layer jump.
ScsiError111 = 465	IsoSDK (see page 1) Error code: Illegal function.
ScsiError112 = 466	IsoSDK (see page 1) Error code: Format-layer may have changed.
ScsiErrorDisk36 = 467	IsoSDK (see page 1) Error code: Cannot write medium - unsupported medium version.
ScsiError113 = 468	IsoSDK (see page 1) Error code: Enclosure services checksum error.
ScsiError114 = 469	IsoSDK (see page 1) Error code: System resource failture.
ScsiError115 = 470	IsoSDK (see page 1) Error code: RMZ extention is not allowed.
ScsiError116 = 471	IsoSDK (see page 1) Error code: No more test zone extensions are allowed.
ScsiError117 = 472	IsoSDK (see page 1) Error code: Current power calibration area is almost full.
ScsiError118 = 473	IsoSDK (see page 1) Error code: Current power calibration area is full.
ScsiError119 = 474	IsoSDK (see page 1) Error code: RDZ is full.
ScsiErrorDcss07 = 475	IsoSDK (see page 1) Error code: Insufficient block count for binding nonce recording.
ScsiErrorDcss08 = 476	IsoSDK (see page 1) Error code: Conflict in binding nonce recording.
SdkMessageWait = 500	IsoSDK (see page 1) Message: Writing lead-out (may take 15-20 minutes).
SdkMessageWritestart = 501	IsoSDK (see page 1) Message: Starting write process.
SdkMessageEreasestart = 502	IsoSDK (see page 1) Message: Starting the erase process.
SdkMessageExtrFile = 503	IsoSDK (see page 1) Message: Extracting file.
SdkMessageSimulate = 504	IsoSDK (see page 1) Message: Test write.
SdkMessageImport = 505	IsoSDK (see page 1) Message: Importing session.
SdkMessageFormat = 506	IsoSDK (see page 1) Message: Formatting medium.
SdkMessageFormatDone = 507	IsoSDK (see page 1) Message: Formatting is done. Thrown after SdkMessageFormat.
SdkMessageImagecreatestart = 508	IsoSDK (see page 1) Message: Image creation started.
SdkMessage01 = 600	IsoSDK (see page 1) Message: Text unknown.
SdkMessage02 = 601	IsoSDK (see page 1) Message: Waiting for user interaction.
SdkMessage03 = 602	IsoSDK (see page 1) Message: Cannot close dialog.
SdkMessage04 = 603	IsoSDK (see page 1) Message: Please stop the process first.
SdkMessage05 = 604	IsoSDK (see page 1) Message: Erasing CD/DVD ...
SdkMessage06 = 605	IsoSDK (see page 1) Message: Preparing Data ...
SdkMessage07 = 606	IsoSDK (see page 1) Message: Finalizing CD/DVD ...
SdkMessage08 = 607	IsoSDK (see page 1) Message: Burning CD/DVD ...
SdkMessage10 = 608	IsoSDK (see page 1) Message: Aborting Process ...

SdkMessage11 = 609	IsoSDK (see page 1) Message: Process successfully completed.
SdkMessage12 = 610	IsoSDK (see page 1) Message: max.
SdkMessage13 = 611	IsoSDK (see page 1) Message: Please insert the next CD/DVD.
SdkMessage14 = 612	IsoSDK (see page 1) Message: Starting the verify process ...
SdkMessage15 = 613	IsoSDK (see page 1) Message: Process completed with %d error(s)
SdkMessage16 = 614	IsoSDK (see page 1) Message: Verifying file.
GuiResource01 = 630	GUI resource string: CD/DVD Eraser depreciated
GuiResource02 = 631	GUI resource string: CD/DVD Burner (see page 13) depreciated
GuiResource03 = 632	GUI resource string: Device depreciated
GuiResource04 = 633	GUI resource string: Progress depreciated
GuiResource05 = 634	GUI resource string: Status depreciated
GuiResource06 = 635	GUI resource string: Fastdepreciated
GuiResource07 = 636	GUI resource string: Complete depreciated
GuiResource08 = 637	GUI resource string: Exit depreciated
GuiResource09 = 638	GUI resource string: Settings depreciated
GuiResource10 = 639	GUI resource string: Start depreciated
GuiResource11 = 640	GUI resource string: Stop depreciated
GuiResource12 = 641	GUI resource string: Simulate burning depreciated
GuiResource13 = 642	GUI resource string: Finalize medium depreciated
GuiResource14 = 643	GUI resource string: Use burnproof depreciated
GuiResource15 = 644	GUI resource string: Speed (see page 201) depreciated
GuiResource16 = 645	GUI resource string: Eject medium after burn depreciated
GuiResource17 = 646	GUI resource string: Joliet file system depreciated
GuiResource18 = 647	GUI resource string: Cache size depreciated
GuiResource19 = 648	GUI resource string: 0.5 MB depreciated
GuiResource20 = 649	GUI resource string: 64 MB depreciated
GuiResource21 = 650	GUI resource string: Medium name depreciated
GuiResource22 = 651	GUI resource string: Make boot medium depreciated
GuiResource23 = 652	GUI resource string: No of Copies depreciated
GuiResource24 = 653	GUI resource string: Medium depreciated
GuiResource25 = 654	GUI resource string: Burning depreciated
GuiResource26 = 655	GUI resource string: General depreciated
GuiResource27 = 656	GUI resource string: Simulation was successful.depreciated
GuiResource28 = 657	GUI resource string: Burn the medium now? depreciated
GuiResource29 = 658	GUI resource string: Verify data after burn depreciated
GuiResource30 = 659	GUI resource string: DVD High-Compatibility-Mode depreciated
GuiResource31 = 660	GUI resource string: Bytes Written: depreciated
GuiResource32 = 661	GUI resource string: Elapsed Time: depreciated
GuiResource33 = 662	GUI resource string: Buffer: depreciated
GuiResource34 = 663	GUI resource string: Remaining Time: depreciated
GuiResource35 = 664	GUI resource string: Other depreciated
GuiResource36 = 665	GUI resource string: Compression depreciated
GuiResource37 = 666	GUI resource string: Encryption depreciated
GuiResource38 = 667	GUI resource string: Password depreciated
GuiResource39 = 668	GUI resource string: UDFdepreciated

GuiResource40 = 669	GUI resource string: Version depreciated
GuiResource41 = 670	GUI resource string: Partition type depreciated
GuiResource42 = 671	GUI resource string: Save Log depreciated
GuiResource43 = 672	GUI resource string: Select multiple devices depreciated
GuiResource44 = 673	GUI resource string: Multiple devices depreciated
GuiResource45 = 674	GUI resource string: Write method depreciated
GuiResource46 = 675	GUI resource string: Confirm depreciated
GuiError01 = 676	GUI error string: Password and confirmation do not match. depreciated
GuiError02 = 677	GUI error string: Password is empty. depreciated
GuiResource47 = 678	GUI resource string: Write streams depreciated
GuiResource48 = 679	GUI resource string: Rescan depreciated
GuiResource49 = 680	GUI resource string: Auto Erase depreciated
GuiResource50 = 681	GUI resource string: Rock Ridge File System depreciated
SdkVerifyErrorHddfileunreadable = 700	IsoSDK (see page 1) Error code: The source file is not readable.
SdkVerifyErrorCdfileunreadable = 701	IsoSDK (see page 1) Error code: The burned file is not readable
SdkVerifyErrorFilesdifferent = 702	IsoSDK (see page 1) Error code: The burned file differs from the source file.

Description

This enumeration defines the error codes the IsoSDK (see page 1) can throw out.

1.1.2.17 IsoSDK::ExtendedDeviceInformation Structure

C++

```
ref struct ExtendedDeviceInformation {
  String ^ Name;
  String ^ Revision;
  int32 RegionCode;
  int32 RegionCodeChangesLeft;
  String ^ LoaderType;
  String ^ ConnectionInterface;
  String ^ PhysicalInterface;
  int32 NumberOfVolumeLevels;
  int32 BufferSize;
  String ^ SerialNumber;
  int32 ReadRetryCount;
  int32 RegionCodeVendorResetsLeft;
  DateTime FirmwareCreationDate;
};
```

C#

```
public struct ExtendedDeviceInformation {
  public String Name;
  public String Revision;
  public int32 RegionCode;
  public int32 RegionCodeChangesLeft;
  public String LoaderType;
  public String ConnectionInterface;
  public String PhysicalInterface;
  public int32 NumberOfVolumeLevels;
  public int32 BufferSize;
  public String SerialNumber;
  public int32 ReadRetryCount;
  public int32 RegionCodeVendorResetsLeft;
  public DateTime FirmwareCreationDate;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ Name;	The full name of the device.
String ^ Revision;	The drive revision description.
int32 RegionCode;	The currently set region code of the device.
int32 RegionCodeChangesLeft;	The available region code changes of the device.
String ^ LoaderType;	The type of the loader, like example “tray”.
String ^ ConnectionInterface;	The interface of the device. Like IDE, SATA, USB and so forth.
String ^ PhysicalInterface;	The physical drive interface like “ATAPI”.
int32 NumberOfVolumeLevels;	The number of volume levels of the device.
int32 BufferSize;	The device buffer size.
String ^ SerialNumber;	The serial number of the device.
int32 ReadRetryCount;	The read retry count that is used by the drive if a read error occur.
int32 RegionCodeVendorResetsLeft;	The available vendor region code changes of the device.
DateTime FirmwareCreationDate;	The firmware date of the device.

Description

A structure that contains the extended information for the device.

1.1.2.18 IsoSDK::ExtendedMediumType Enumeration

C++

```
enum class ExtendedMediumType {
  CdRom = 0,
  CdRomXA = 1,
  CdAudio = 2,
  CdMixedMode = 3,
  CdEnhanced = 4,
  CdMultisession = 5,
  Dvd = 6,
  Bd = 7,
  HdDvd = 8
};
```

C#

```
public enum ExtendedMediumType {
  CdRom = 0,
  CdRomXA = 1,
  CdAudio = 2,
  CdMixedMode = 3,
  CdEnhanced = 4,
  CdMultisession = 5,
  Dvd = 6,
  Bd = 7,
  HdDvd = 8
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
CdRom = 0	Medium type: Data CD or CD of unknown format.
CdRomXA = 1	Medium type: CD-eXtended Architecture format.
CdAudio = 2	Medium type: CD-DA format.
CdMixedMode = 3	Medium type: Mixed Mode CD format.
CdEnhanced = 4	Medium type: CD-Enhanced format.
CdMultisession = 5	Medium type: Multi-session data CD.
Dvd = 6	Medium type: DVD medium.
Bd = 7	Medium type: BluRay medium.
HdDvd = 8	Medium type: HDDVD medium.

Description

This enumeration defines the extended type of the current medium.

1.1.2.19 IsoSDK::Extent Structure

C++

```
ref struct Extent {
  int Location;
  int Length;
};
```

C#

```
public struct Extent {
  public int Location;
  public int Length;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
int Location;	LBA of the first sector of the file extent.
int Length;	Size of the file extent in bytes.

Description

A structure that contains the information about the Extent information of a file in allocation table.

1.1.2.20 IsoSDK::FileAllocationTable Structure

C++

```
ref struct FileAllocationTable {
  short EmbeddedFileOffset;
  array<Extent ^> ^ Extents;
};
```

C#

```
public struct FileAllocationTable {
  public short EmbeddedFileOffset;
  public array<Extent ^> Extents;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
short EmbeddedFileOffset;	If it is not 0 it contains the offset in bytes of the file data in File Entry (file descriptor) and it will have exact one extend info with the file size and location. If this is 0 there will be an array of extents available.
array<Extent ^> ^ Extents;	An array with the available Extent (see page 183) information.

Description

A structure that contains the information about the file allocation table.

1.1.2.21 IsoSDK::FileAttributes Enumeration

C++

```
[Flags]
enum class FileAttributes {
  ReadOnly = 0x001,
  Hidden = 0x002,
  System = 0x004,
  Directory = 0x010,
  Archive = 0x020,
  AdvancedHidden = 0x040,
  All = 0x07F
};
```

C#

```
[Flags]
public enum FileAttributes {
  ReadOnly = 0x001,
  Hidden = 0x002,
  System = 0x004,
  Directory = 0x010,
  Archive = 0x020,
  AdvancedHidden = 0x040,
  All = 0x07F
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
ReadOnly = 0x001	Write protected files.
Hidden = 0x002	File is hidden.
System = 0x004	File is a System file.
Directory = 0x010	Directory / Folder.
Archive = 0x020	File is an archive file.
AdvancedHidden = 0x040	File attribute: Advanced hidden files. Not valid in all OS. Use in combination with ISO files.
All = 0x07F	Use all attributes for the file.

Description

This enumeration defines the file attribute of a file to add.

1.1.2.22 IsoSDK::FileEntry Structure

C++

```
ref struct FileEntry {
  String ^ Name;
  String ^ Path;
  String ^ Origin;
  int32 Address;
  __int64 Size;
  FileDateTime ^ DateTime;
  FileAttributes Attributes;
  FileDateTime ^ CreationTime;
  FileDateTime ^ AccessTime;
  void * UserParam;
};
```

C#

```
public struct FileEntry {
  public String Name;
  public String Path;
  public String Origin;
  public int32 Address;
  public __int64 Size;
  public FileDateTime DateTime;
  public FileAttributes Attributes;
  public FileDateTime CreationTime;
  public FileDateTime AccessTime;
  public void * UserParam;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ Name;	The name of the directory entry.
String ^ Path;	The path to the containing directory of this entry.
String ^ Origin;	The path of the file on HDD. If the file is imported from the disk session, this field is NULL.
int32 Address;	This is the file descriptor address of the file (LBA).
__int64 Size;	The size of the file in bytes.
FileDateTime ^ DateTime;	Creation date and time of the current entry according to a FileDateTime (see page 101) structure.
FileAttributes Attributes;	An attribute value according to the FileAttributes (see page 184) enumeration.
FileDateTime ^ CreationTime;	The creation date of the file according the file system. Use a FileAttributes (see page 184) structure as value.
FileDateTime ^ AccessTime;	The last access date of the file according the file system. Use a FileDateTime (see page 101) structure as value.
void * UserParam;	This is a data value you can set yourself according to your needs. You can pass an pointer to a integer or structure to the file that stores extra data.

Description

A structure that contains the information for the file to be used.

1.1.2.23 IsoSDK::FileSystems Enumeration

C++

```
enum class FileSystems {
  Unknown = 0,
  Iso9660 = 1,
  Joliet = 2,
  Udf = 4,
  Bootable = 8,
  RockRidge = 16
};
```

C#

```
public enum FileSystems {
  Unknown = 0,
  Iso9660 = 1,
  Joliet = 2,
  Udf = 4,
  Bootable = 8,
  RockRidge = 16
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Unknown = 0	Disk file system: No file system or unknown file system.
Iso9660 = 1	Disk file system: ISO 9660.
Joliet = 2	Disk file system: Joliet extension to ISO 9660.
Udf = 4	Disk file system: Universal Disk Format.
Bootable = 8	Disk file system: El Torito bootable extension to ISO 9660.
RockRidge = 16	Disk file system: Rockridge

Description

This enumeration defines the file system of a medium the IsoSDK (see page 1) supports.

1.1.2.24 IsoSDK::ImageFormat Enumeration

C++

```
enum class ImageFormat {
  Iso = 1,
  Bin = 2
};
```

C#

```
public enum ImageFormat {
  Iso = 1,
  Bin = 2
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Iso = 1	FileFormat is ISO (2048).

Bin = 2	FileFormat is BIN (2352).

Description

This enumeration defines the image formats the IsoSDK (see page 1) supports for writing.

1.1.2.25 IsoSDK::ImageTask Enumeration

C++

```
enum class ImageTask {
  Create = 1,
  Verify = 2,
  CreateVerify = 3
};
```

C#

```
public enum ImageTask {
  Create = 1,
  Verify = 2,
  CreateVerify = 3
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Create = 1	The IsoSDK (see page 1) will create a ImageFile.
Verify = 2	The IsoSDK (see page 1) will verify a ImageFile.
CreateVerify = 3	The IsoSDK (see page 1) will create and verify a ImageFile.

Description

This enumeration defines the possible actions for image creation task while disk copy.

1.1.2.26 IsoSDK::InfoLevel Enumeration

C++

```
enum class InfoLevel {
  Info = 0,
  LowDebug = 1,
  MediumDebug = 2,
  HighDebug = 3
};
```

C#

```
public enum InfoLevel {
  Info = 0,
  LowDebug = 1,
  MediumDebug = 2,
  HighDebug = 3
}
```

File

EventArgs.h

Members

Members	Description
Info = 0	Loginfo Level: Info. Use this for your own log files.

LowDebug = 1	Loginfo Level: Low. Use only for debugging.
MediumDebug = 2	Loginfo Level: Medium. Use only for debugging.
HighDebug = 3	Loginfo Level: High. Use only for debugging.

Description

This enumeration defines the level of informations that the IsoSDK (see page 1) supports.

1.1.2.27 IsoSDK::ISOLevel Enumeration

C++

```
enum class ISOLevel {
  Level1 = 1,
  Level2 = 2,
  Level3 = 3,
  Romeo = 4
};
```

C#

```
public enum ISOLevel {
  Level1 = 1,
  Level2 = 2,
  Level3 = 3,
  Romeo = 4
}
```

File

Options.h

Members

Members	Description
Level1 = 1	Disk file system: ISO Level 1 Specification: 1) Name format is 8.3. Available chars: only capital letters and underscore. 2) File extentio length is limited to 3 chars. 3) Directory can't has extention. 4) File size limit is 2 GB.
Level2 = 2	Disk file system: ISO Level 2 Specification: 1) Max name length is 31 chars. 2) File size limit is 2 GB.
Level3 = 3	Disk file system: ISO Level 3 Specification: 1) Max name length is 128 chars. 2) No file size limit.
Romeo = 4	Disk file system: ISO Level Romeo Specification: 1) Max name length is 128 chars. 2) Lower case chars are allowed in file names.

Description

This enumeration defines the possible extended ISO9660 derivate the IsoSDK (see page 1) supports.

1.1.2.28 IsoSDK::ISOVolumeInfo Structure

C++

```
ref struct ISOVolumeInfo {
  String ^ VolumeLabel;
  long VolumeDescriptorAddress;
  long VolumeSize;
  long RootAddress;
  long PathTableAddress;
  long PathTableSize;
  String ^ AbstractFileIdentifier;
  String ^ ApplicationIdentifier;
  String ^ BiblioIdentifier;
  String ^ CopyrightFileIdentifier;
  String ^ DataPreparerIdentifier;
  String ^ PublisherIdentifier;
  String ^ SetIdentifier;
  String ^ SystemIdentifier;
  FileDateTime ^ RootDateTime;
  FileDateTime ^ CreationDateTime;
  FileDateTime ^ ModificationDateTime;
  FileDateTime ^ ExpirationDateTime;
  FileDateTime ^ EffectiveDateTime;
};
```

C#

```
public struct ISOVolumeInfo {
  public String VolumeLabel;
  public long VolumeDescriptorAddress;
  public long VolumeSize;
  public long RootAddress;
  public long PathTableAddress;
  public long PathTableSize;
  public String AbstractFileIdentifier;
  public String ApplicationIdentifier;
  public String BiblioIdentifier;
  public String CopyrightFileIdentifier;
  public String DataPreparerIdentifier;
  public String PublisherIdentifier;
  public String SetIdentifier;
  public String SystemIdentifier;
  public FileDateTime RootDateTime;
  public FileDateTime CreationDateTime;
  public FileDateTime ModificationDateTime;
  public FileDateTime ExpirationDateTime;
  public FileDateTime EffectiveDateTime;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ VolumeLabel;	The volume label name.
long VolumeDescriptorAddress;	The address (sector) where the volume descriptor starts.
long VolumeSize;	The size of the disk in sectors.
long RootAddress;	The root address of the ISO filesystem.
long PathTableAddress;	The address where the path table starts.
long PathTableSize;	The size of the path table in bytes.
String ^ AbstractFileIdentifier;	The Volume Abstract File Identifier field of the ISO/Joliet image (35 characters maximum). For detailed information see: ISO 9660

String ^ ApplicationIdentifier;	The Volume Application Identifier field of the ISO/Joliet image(128 characters maximum). For detailed information see: ISO 9660
String ^ BiblioIdentifier;	The Volume Bibliography File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660
String ^ CopyrightFileIdentifier;	Volume Copyright File Identifier field of the ISO/Joliet image (36 characters maximum). For detailed information see: ISO 9660
String ^ DataPreparerIdentifier;	The Volume Data Preparer Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660
String ^ PublisherIdentifier;	The Volume Publisher Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660
String ^ SetIdentifier;	The Volume Set Identifier field of the ISO/Joliet image (128 characters maximum). For detailed information see: ISO 9660
String ^ SystemIdentifier;	The Volume System Identifier field of the ISO/Joliet image (31 characters maximum). For detailed information see: ISO 9660
FileDateTime ^ RootDateTime;	The date / time info when the disc or images was created. You can overwrite it with FileDateTimeEx (see page 104).
FileDateTime ^ CreationDateTime;	This is the date / time information that was set with IsoExOptions. if not set this was "CurrentTime" while creation.
FileDateTime ^ ModificationDateTime;	This is the date / time information that was set with IsoExOptions. if not set this was "CurrentTime" while creation.
FileDateTime ^ ExpirationDateTime;	This is the date / time information that was set with IsoExOptions. if not set this is set to default date 1970-1-1 00:00:00
FileDateTime ^ EffectiveDateTime;	This is the date / time information that was set with IsoExOptions. if not set this is set to default date 1970-1-1 00:00:00

Description

A structure that contains the information about the ISO information.

1.1.2.29 IsoSDK::MediumInfo Structure

C++

```
ref struct MediumInfo {
  int FirstSession;
  double FreeSize;
  SessionStatus LastSesionStatus;
  int LastSession;
  int FirstTrack;
  int LastTrack;
  double Size;
  MediumStatus Status;
  String ^ Type;
  MediumType TypeCode;
  double UsedSize;
  ExtendedMediumType ExtendedType;
  String ^ UPCEANCode;
  String ^ VendorID;
  float MaxWriteSpeed;
};
```

C#

```
public struct MediumInfo {
  public int FirstSession;
  public double FreeSize;
  public SessionStatus LastSesionStatus;
  public int LastSession;
  public int FirstTrack;
  public int LastTrack;
  public double Size;
  public MediumStatus Status;
  public String Type;
  public MediumType TypeCode;
  public double UsedSize;
  public ExtendedMediumType ExtendedType;
  public String UPCEANCode;
  public String VendorID;
  public float MaxWriteSpeed;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
int FirstSession;	The number of the first session of the current medium.
double FreeSize;	The available space of the medium in bytes.
SessionStatus LastSesionStatus;	The status of the last session according to the SessionStatus (see page 201) enumeration.
int LastSession;	The number of the last session of the current medium.
int FirstTrack;	The number of the first track of the current medium.
int LastTrack;	The number of the last track of the current medium.
double Size;	The current medium size in bytes.
MediumStatus Status;	The status of the current medium according to the MediumStatus (see page 191) enumeration.
String ^ Type;	Returns the medium type in clear text.
MediumType TypeCode;	The type of the current medium according to the MediumType (see page 192) enumeration.
double UsedSize;	Already used space of the current medium in bytes.
ExtendedMediumType ExtendedType;	The type of the current medium according to the ExtendedMediumType (see page 182) enumeration.
String ^ UPCEANCode;	Returns the UPCEAN Code of the current medium.
String ^ VendorID;	The vendor ID of the disk producer.
float MaxWriteSpeed;	This is the max. write speed the drive reported. This is not supported by all drives.

Description

A structure that contains the information about the medium.

1.1.2.30 IsoSDK::MediumStatus Enumeration

C++

```
enum class MediumStatus {
  EmptyDisk = 0,
  IncompleteDisk = 1,
  CompleteDisk = 2,
  Other = 3
};
```

C#

```
public enum MediumStatus {
  EmptyDisk = 0,
  IncompleteDisk = 1,
  CompleteDisk = 2,
  Other = 3
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
EmptyDisk = 0	Disk status: Disk is empty.
IncompleteDisk = 1	Disk status: Disk is incomplete (not finalized).
CompleteDisk = 2	Disk status: Disk is complete / finalized.
Other = 3	Disk status: Unknown

Description

This enumeration defines the status of the current medium.

1.1.2.31 IsoSDK::MediumType Enumeration

C++

```
enum class MediumType {
  Unknown = 0,
  CdRom = 1,
  CdR = 2,
  CdRw = 3,
  DvdRom = 4,
  DvdR = 5,
  DvdRam = 6,
  DvdRwRo = 7,
  DvdRw = 8,
  DvdRwSr = 9,
  DvdPlusRw = 10,
  DvdPlusR = 11,
  DdCdRom = 12,
  DdCdR = 13,
  DdCdRw = 14,
  DvdRDLPlus = 15,
  DvdRwDLPlus = 16,
  DvdMrDL = 17,
  BlurayR = 18,
  BlurayRe = 19,
  BlurayRom = 20,
  BlurayRRrm = 21,
  HdDvdR = 22,
  HdDvdRw = 23,
  HdDvdRom = 24,
  HdDvdRam = 25,
  HdDvdRDL = 26,
  HdDvdRwDL = 27
};
```

C#

```
public enum MediumType {
  Unknown = 0,
  CdRom = 1,
  CdR = 2,
  CdRw = 3,
  DvdRom = 4,
  DvdR = 5,
```

```
    DvdRam = 6,
    DvdRwRo = 7,
    DvdRw = 8,
    DvdRwSr = 9,
    DvdPlusRw = 10,
    DvdPlusR = 11,
    DdCdRom = 12,
    DdCdR = 13,
    DdCdRw = 14,
    DvdRDLPlus = 15,
    DvdRwDLPlus = 16,
    DvdMrDL = 17,
    BlurayR = 18,
    BlurayRe = 19,
    BlurayRom = 20,
    BlurayRRrm = 21,
    HdDvdR = 22,
    HdDvdRw = 23,
    HdDvdRom = 24,
    HdDvdRam = 25,
    HdDvdRDL = 26,
    HdDvdRwDL = 27
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Unknown = 0	A disk identifier for disk type: Unknown disk type.
CdRom = 1	A disk identifier for disk type: CD-ROM.
CdR = 2	A disk identifier for disk type: CD-R.
CdRw = 3	A disk identifier for disk type: CD-ReWriteable.
DvdRom = 4	A disk identifier for disk type: DVD-ROM.
DvdR = 5	A disk identifier for disk type: DVD-R.
DvdRam = 6	A disk identifier for disk type: DVD-RAM.
DvdRwRo = 7	A disk identifier for disk type: DVD-R ReWriteable - Restricted Overwrite.
DvdRw = 8	A disk identifier for disk type: DVD-R ReWriteable.
DvdRwSr = 9	A disk identifier for disk type: DVD-R ReWriteable - Sequential Recording.
DvdPlusRw = 10	A disk identifier for disk type: DVD+R ReWriteable.
DvdPlusR = 11	A disk identifier for disk type: DVD+R.
DdCdRom = 12	A disk identifier for disk type: Double Density Compact Disk - ROM.
DdCdR = 13	A disk identifier for disk type: Double Density Compact Disk - Recording.
DdCdRw = 14	A disk identifier for disk type: Double Density Compact Disk - ReWriteable.
DvdRDLPlus = 15	A disk identifier for disk type: DVD+R Doube Layer.
DvdRwDLPlus = 16	A disk identifier for disk type: DVD+R ReWriteable Double Layer.
DvdMrDL = 17	A disk identifier for disk type: DVD-R Double Layer.
BlurayR = 18	A disk identifier for disk type: Blu-ray BD-R.
BlurayRe = 19	A disk identifier for disk type: Blu-ray BD-RE.
BlurayRom = 20	A disk identifier for disk type: Blu-ray BD-ROM.
BlurayRRrm = 21	A disk identifier for disk type: Blu-ray BD-RRM.
HdDvdR = 22	A disk identifier for disk type: HDDVD R.
HdDvdRw = 23	A disk identifier for disk type: HDDVD-ReWriteable.

HdDvdRom = 24	A disk identifier for disk type: HDDVD-ROM.
HdDvdRam = 25	A disk identifier for disk type: HDDVD-RAM.
HdDvdRDL = 26	A disk identifier for disk type: HDDVD R Double Layer.
HdDvdRwDL = 27	A disk identifier for disk type: HDDVD ReWriteable Double Layer.

Description

This enumeration defines the type of the current medium.

1.1.2.32 IsoSDK::NetworkTagsContentItem Enumeration

C++

```
enum class NetworkTagsContentItem {
  Category = 0,
  DiskId = 1,
  Artist = 2,
  Title = 3,
  SubmittedVIA = 4,
  Genre = 5,
  ExtendedInfo = 6,
  DiskLengrh = 7,
  FrameOffset = 8,
  Revision = 9,
  Year = 10
};
```

C#

```
public enum NetworkTagsContentItem {
  Category = 0,
  DiskId = 1,
  Artist = 2,
  Title = 3,
  SubmittedVIA = 4,
  Genre = 5,
  ExtendedInfo = 6,
  DiskLengrh = 7,
  FrameOffset = 8,
  Revision = 9,
  Year = 10
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Category = 0	AudioDisk CDDB Tag Information: Category Information.
DiskId = 1	AudioDisk CDDB Tag Information: Get the DiskID.
Artist = 2	AudioDisk CDDB Tag Information: Artist Information.
Title = 3	AudioDisk CDDB Tag Information: Receive Track or Disk Title.
SubmittedVIA = 4	AudioDisk CDDB Tag Information: Receive Information about the identifier that was used to submit data.
Genre = 5	AudioDisk CDDB Tag Information: Genre Information.
ExtendedInfo = 6	AudioDisk CDDB Tag Information: Extended Information.
DiskLengrh = 7	AudioDisk CDDB Tag Information: Disk Length Information.
FrameOffset = 8	AudioDisk CDDB Tag Information: Frame Offset Information.
Revision = 9	AudioDisk CDDB Tag Information: Revision Information.
Year = 10	AudioDisk CDDB Tag Information: Receive Production Year.

Description

This enumeration defines the CDDB fields that are supported by the IsoSDK (see page 1).

1.1.2.33 IsoSDK::ProjectType Enumeration

C++

```
enum class ProjectType {
  Audio = 0,
  Cue = 1,
  Data = 2,
  Vcd = 3,
  Svcd = 4,
  VideoDvd = 5,
  UdfDvd = 6,
  IsoUdf = 7,
  Bluray = 8,
  MixedMode = 9,
  Raw = 10
};
```

C#

```
public enum ProjectType {
  Audio = 0,
  Cue = 1,
  Data = 2,
  Vcd = 3,
  Svcd = 4,
  VideoDvd = 5,
  UdfDvd = 6,
  IsoUdf = 7,
  Bluray = 8,
  MixedMode = 9,
  Raw = 10
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Audio = 0	Project type Audio CD.
Cue = 1	Project type Bin/Cue image.
Data = 2	Project type Data CD/DVD.
Vcd = 3	Project type Video CD (Mpeg1).
Svcd = 4	Project type Super Video CD (Mpeg2).
VideoDvd = 5	Project type Video DVD (ISO/UDF 1.02).
UdfDvd = 6	Project type Data CD/DVD with UDF only.
IsoUdf = 7	Project type Data CD/DVD with ISO UDF bridge
Bluray = 8	Project type Blu-ray disk.
MixedMode = 9	Project type Mixed Mode CD (audio/data).
Raw = 10	Project type RAW image (Universal).

Description

This enumeration defines the project types the IsoSDK (see page 1) supports.

1.1.2.34 IsoSDK::RawDataType Enumeration

C++

```
enum class RawDataType {
  No = 0x0000,
  SyncHeader = 0x0001,
  SubHeaders = 0x0002,
  Data = 0x0004,
  EdcEcc = 0x0008,
  SubchPQ = 0x0010,
  SubchPW = 0x0020,
  SubchRW = 0x0040
};
```

C#

```
public enum RawDataType {
  No = 0x0000,
  SyncHeader = 0x0001,
  SubHeaders = 0x0002,
  Data = 0x0004,
  EdcEcc = 0x0008,
  SubchPQ = 0x0010,
  SubchPW = 0x0020,
  SubchRW = 0x0040
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
No = 0x0000	For Ignore Data Mask to be empty.
SyncHeader = 0x0001	Input data contains the sync/header field.
SubHeaders = 0x0002	Input data contains the subheaders field.
Data = 0x0004	Input data contains UserData field.
EdcEcc = 0x0008	Input data contains ECC/EDC fields.
SubchPQ = 0x0010	Input data contains P-Q subchannel data in packed form.
SubchPW = 0x0020	Input data contains P-W subchannel data in raw form.
SubchRW = 0x0040	Input data contains R-W subchannel data in raw form.

Description

This enumeration defines the data type for a RAW Project type.

1.1.2.35 IsoSDK::RawTrack Structure

C++

```
ref struct RawTrack {
  int32 Number;
  int32 Index;
  RawTrackFormat Format;
  RawDataType DataTypeMask;
  RawDataType IgnoreDataMask;
  int32 StartAddress;
  int32 Length;
  int64 Offset;
};
```

C#

```
public struct RawTrack {
```

```
    public int32 Number;
    public int32 Index;
    public RawTrackFormat Format;
    public RawDataType DataTypeMask;
    public RawDataType IgnoreDataMask;
    public int32 StartAddress;
    public int32 Length;
    public int64 Offset;
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
int32 Number;	The number of the track. 0 - Lead-in, 0xAA – lead-out. 1 – first track, 2 – second track ...
int32 Index;	Index inside the track. 0 - pre-gap, 1..99 - user data
RawTrackFormat Format;	The format of the track according to the RawTrackFormat (see page 197) enumeration.
RawDataType DataTypeMask;	The format of the data according to the RawDataType (see page 196) enumeration.
RawDataType IgnoreDataMask;	The format of the data to ignore according to the RawDataType (see page 196) enumeration.
int32 StartAddress;	This is the start address of the track on the disk in sectors.
int32 Length;	The length of the track in sectors.
int64 Offset;	The offset of the data in the image file in bytes. Not used if burning through callback function.

Description

This structure contains the information of a RAW image.

Remarks

Create this structure as a Array to create a list of structures for a disk structure.

Example

```
  private List<RawTrack> m_TrackList = new List<RawTrack>();
  RawTrack rawTrack = new RawTrack();
  rawTrack.Index = nIndex;
  ....
  m_TrackList.Add(rawTrack);
```

1.1.2.36 IsoSDK::RawTrackFormat Enumeration

C++

```
  enum class RawTrackFormat {
    Audio = 0,
    Mode1 = 1,
    Mode2Formless = 2,
    Mode2Form1 = 3,
    Mode2Form2 = 4
  };
```

C#

```
  public enum RawTrackFormat {
    Audio = 0,
    Mode1 = 1,
    Mode2Formless = 2,
    Mode2Form1 = 3,
    Mode2Form2 = 4
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Audio = 0	Audio track.
Mode1 = 1	Mode 1 track.
Mode2Formless = 2	Mode 2 formless track.
Mode2Form1 = 3	Mode 2 Form 1 track.
Mode2Form2 = 4	Mode 2 Form 2 track.

Description

This enumeration defines the track format type for a RAW project type.

1.1.2.37 IsoSDK::ReadErrorCorrectionParams Structure

C++

```
ref struct ReadErrorCorrectionParams {
  bool ErrorCorrection;
  bool BlankBadSectors;
  int HardwareRetryCount;
  int SoftwareRetryCount;
};
```

C#

```
public struct ReadErrorCorrectionParams {
  public bool ErrorCorrection;
  public bool BlankBadSectors;
  public int HardwareRetryCount;
  public int SoftwareRetryCount;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
bool ErrorCorrection;	A bool value that indicates to use the software and hardware retry functions.
bool BlankBadSectors;	A bool value that indicate to fill BadSectors with blank sectors on the target disk or image. This will help to keep the original size and positions.
int HardwareRetryCount;	This value indicates how many times the device try to read possible bad sectors.
int SoftwareRetryCount;	This value indicates how many times the software try to read possible bad sectors.

Description

A structure that contains the information of the ReadErrorCorrection for CopyDisk and CreateImage.

1.1.2.38 IsoSDK::ReadMode Enumeration

C++

```
enum class ReadMode {
  Iso = 0,
```

```
  Raw = 1,
  RawSubchannel = 2
};
```

C#

```
public enum ReadMode {
  Iso = 0,
  Raw = 1,
  RawSubchannel = 2
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Iso = 0	IsoSDK (see page 1) ReadMode: Userdata (2048)
Raw = 1	IsoSDK (see page 1) ReadMode: RAW (2352)
RawSubchannel = 2	IsoSDK (see page 1) ReadMode: RAW Subchannel (2352 +16)

Description

This enumeration defines the read mode the IsoSDK (see page 1) supports.

1.1.2.39 IsoSDK::SavePathOption Enumeration

C++

```
enum class SavePathOption {
  DontSavePath = 0,
  WholePath = 1,
  ParentOnly = 2
};
```

C#

```
public enum SavePathOption {
  DontSavePath = 0,
  WholePath = 1,
  ParentOnly = 2
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
DontSavePath = 0	The file is created in the destination path of the project.
WholePath = 1	The complete path of the file is created as a sub folder in the destination path of the project.
ParentOnly = 2	Only the superordinate directory of the file will be created in destination path of the project. Example: chSourceDir = c:\tmp1\tmp2\test.tmp - only the sub folder tmp2 will be created.

Description

This enumeration defines if and how the super ordinate directory is added as a folder to the project.

1.1.2.40 IsoSDK::SaveTrackFileFormat Enumeration

C++

```
enum class SaveTrackFileFormat {
  Wave = 0,
  Iso = 1,
  Bin = 2,
  Mpeg = 3
};
```

C#

```
public enum SaveTrackFileFormat {
  Wave = 0,
  Iso = 1,
  Bin = 2,
  Mpeg = 3
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Wave = 0	File format is WAVE (PCM). Audio file.
Iso = 1	File format is ISO (2048).
Bin = 2	File format is BIN (2352).
Mpeg = 3	FileFormat is MPEG video File.

Description

This enumeration defines the possible formats the IsoSDK (see page 1) supports to save tracks.

1.1.2.41 IsoSDK::SessionInfo Structure

C++

```
ref struct SessionInfo {
  bool LastSession;
  long Size;
  long StartLBA;
  long FirstTrack;
  long LastTrack;
};
```

C#

```
public struct SessionInfo {
  public bool LastSession;
  public long Size;
  public long StartLBA;
  public long FirstTrack;
  public long LastTrack;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
bool LastSession;	Is it the last disk's session.
long Size;	Session size in sectors.

long StartLBA;	Address of the first session's sector.
long FirstTrack;	Number of the first track of the session.
long LastTrack;	Number of the last track of the session.

Description

A structure that contains the information about the selected session.

1.1.2.42 IsoSDK::SessionStatus Enumeration

C++

```
enum class SessionStatus {
  EmptySession = 0,
  IncompleteSession = 1,
  DamagedSession = 2,
  CompleteSession = 3
};
```

C#

```
public enum SessionStatus {
  EmptySession = 0,
  IncompleteSession = 1,
  DamagedSession = 2,
  CompleteSession = 3
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
EmptySession = 0	Disk session status: Session is empty.
IncompleteSession = 1	Disk session status: Session is incomplete.
DamagedSession = 2	Disk session status: Session is damaged.
CompleteSession = 3	Disk session status: Complete.

Description

This enumeration defines the status of the last session.

1.1.2.43 IsoSDK::Speed Structure

C++

```
ref struct Speed {
  float SpeedInX;
  int SpeedInKBPerSec;
};
```

C#

```
public struct Speed {
  public float SpeedInX;
  public int SpeedInKBPerSec;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
float SpeedInX;	Values of normal speed information, like 32.0 or 48.0
int SpeedInKBPerSec;	The original internal speed, like 4800 (32x) or 7200 (48x).

Description

This structure contains information about the possible burning speeds.

1.1.2.44 IsoSDK::TagChoiceType Enumeration

C++

```
enum class TagChoiceType {
  None = 0,
  CdText = 1,
  FreeDb = 2,
  CdText_FreeDb = 3,
  FreeDb_CdText = 4
};
```

C#

```
public enum TagChoiceType {
  None = 0,
  CdText = 1,
  FreeDb = 2,
  CdText_FreeDb = 3,
  FreeDb_CdText = 4
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
None = 0	Audiograbber Tag Information: Receive no tags.
CdText = 1	Audiograbber Tag Information: Receive tags only from CD-Text.
FreeDb = 2	Audiograbber Tag Information: Receive tags only from FreeDB / CDDB.
CdText_FreeDb = 3	Audiograbber Tag Information: Receive tags first from CD-Text then from FreeDB / CDDB.
FreeDb_CdText = 4	Audiograbber Tag Information: Receive tags First from FreeDB/CDDB then from CD-Text.

Description

This enumeration defines the type of tags the IsoSDK (see page 1) will try to receive.

1.1.2.45 IsoSDK::TrackFormat Enumeration

C++

```
enum class TrackFormat {
  Audio = 0,
  DataMode1 = 1,
  DataMode2 = 2
};
```

C#

```
public enum TrackFormat {
```

```
    Audio = 0,
    DataMode1 = 1,
    DataMode2 = 2
  }
```

File

IsoSDKBurnerNet.h

Members

Members	Description
Audio = 0	Track type: Audio track
DataMode1 = 1	Track type: Mode 1 track (always for DVD/BD)
DataMode2 = 2	Track type: Mode 2 track

Description

This enumeration defines the track type of a medium.

1.1.2.46 IsoSDK::TrackInfo Structure

C++

```
ref struct TrackInfo {
  int TrackNumber;
  int SessionNumber;
  long StartLBA;
  long Size;
  TrackFormat Format;
  FileSystems FileSystem;
};
```

C#

```
public struct TrackInfo {
  public int TrackNumber;
  public int SessionNumber;
  public long StartLBA;
  public long Size;
  public TrackFormat Format;
  public FileSystems FileSystem;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
int TrackNumber;	Track number, index of the current medium.
int SessionNumber;	Session number of track.
long StartLBA;	Track's start address.
long Size;	The track size in bytes.
TrackFormat Format;	The format of the track according to the TrackFormat (see page 202) enumeration.
FileSystems FileSystem;	The file system of the track according to the FileSystems (see page 186) enumeration.

Description

A structure that contains the information about the selected track.

1.1.2.47 IsoSDK::UDFPartitionType Enumeration

C++

```
enum class UDFPartitionType {
  Physical = 0,
  Virtual = 1,
  Sparable = 2
};
```

C#

```
public enum UDFPartitionType {
  Physical = 0,
  Virtual = 1,
  Sparable = 2
}
```

File

Options.h

Members

Members	Description
Physical = 0	UDF partition type: Physical or Type 1 This is the simplest partition. A type 1 partition has a start address S and size N. A logical block number A in the partition can be converted to the media physical address (in UDF's term, the logical sector address) S+A. In certain optical media, the start and size of the partition must be aligned to the packet size (such as 32KB). These special requirements are defined in the appendixes of the UDF standard. Free space of the partition is managed by the Unallocated Space Bitmap Descriptor. It contains one bit for each block of the partition. If the bit is set (1), the corresponding block is free. If it is clear (0), the corresponding block is allocated. The is contrary to what FFS/UFS uses the bitmap, because the bitmap in UDF is called Unallocated Space Bitmap.
Virtual = 1	UDF partition type: Virtual. Virtual partition is used on write-once media. Only three types of metadata are stored in the virtual partition: File Set Descriptor, File Entry (including Extended File Entry), and Allocation Extent (see page 183) Descriptor. If the file data is embedded in the file entry, these file data are also stored in the virtual partition. Virtual partition makes the write-once media appear as an overwrite media. Virtual partition layers on top of the type 1 partition. A Virtual Allocation Table (VAT) is used to map logical addresses of the virtual partition to logical addresses in the underlying type 1 partition.

Sparable = 2	UDF partition type: Sparable. Sparable partitions are used on overwrite media that will fail after a certain number of overwrites (several thousands), such as CD-RW. In a file system, the places that are overwritten frequently are often important metadata area, e.g., bitmaps. Sparable partition allows the failed area to be remapped to other good part on the media so the failed area appears good to the upper level. A sparable partition is similar to a type 1 partition in the sense that it has a start address and size. Moreover, it defines 2 to 4 sparing tables which points to reserved spare area on the media. Each sparing table has identical information. The unit of overwrite on such media is packet. For example, the packet size for CD-RW is 32 2K-sectors. One sector in packet failing means the whole packet fails. When this happens, the content of this packet is written to a spare area, and its new address is written to the sparing table. When translating a logical address in the sparable partition to the physical address, the sparing table is always consulted. If the logical address is not found in the sparing table, the address translation is the same as a type 1 partition. Otherwise, its new address in the sparing area recorded in the sparing table is returned. Thus, the sparing table acts as an exception table in the address translation. This mechanism guarantees that the logical address does not change when its original packet fails.

Description

This enumeration defines the UDF partition the IsoSDK (see page 1) supports.

1.1.2.48 IsoSDK::UDFVersion Enumeration

C++

```
enum class UDFVersion {
  Udf102 = 0x102,
  Udf150 = 0x150,
  Udf200 = 0x200,
  Udf201 = 0x201,
  Udf250 = 0x250,
  Udf260 = 0x260
};
```

C#

```
public enum UDFVersion {
  Udf102 = 0x102,
  Udf150 = 0x150,
  Udf200 = 0x200,
  Udf201 = 0x201,
  Udf250 = 0x250,
  Udf260 = 0x260
}
```

File

Options.h

Members

Members	Description
Udf102 = 0x102	UDF Version: 1.02 (VideoDVD)
Udf150 = 0x150	UDF Version: 1.50
Udf200 = 0x200	UDF Version: 2.00
Udf201 = 0x201	UDF Version: 2.01

Udf250 = 0x250	UDF Version: 2.50
Udf260 = 0x260	UDF Version: 2.60

Description

This enumeration defines the UDF version the IsoSDK (see page 1) supports.

1.1.2.49 IsoSDK::UDFVolumeInfo Structure

C++

```
ref struct UDFVolumeInfo {
  String ^ VolumeLabel;
  String ^ Preparer;
  UDFVersion Version;
  UDFPartitionType PartitionType;
  long MVDSAddress;
  long RVDSAddress;
  long RootAddress;
  long RootFEAddress;
  long PartitionAddress;
  long PartitionLength;
  long LVDAddress;
  long PVDAddress;
  long FSDAddress;
  long VATAddress;
  long MetadataAddress;
  long SparingAddress;
  long FileCount;
  long DirCount;
  FileDateTime ^ RecordingDateTime;
};
```

C#

```
public struct UDFVolumeInfo {
  public String VolumeLabel;
  public String Preparer;
  public UDFVersion Version;
  public UDFPartitionType PartitionType;
  public long MVDSAddress;
  public long RVDSAddress;
  public long RootAddress;
  public long RootFEAddress;
  public long PartitionAddress;
  public long PartitionLength;
  public long LVDAddress;
  public long PVDAddress;
  public long FSDAddress;
  public long VATAddress;
  public long MetadataAddress;
  public long SparingAddress;
  public long FileCount;
  public long DirCount;
  public FileDateTime RecordingDateTime;
}
```

File

IsoSDKBurnerNet.h

Members

Members	Description
String ^ VolumeLabel;	The name of the UDF volume label.
String ^ Preparer;	The name of the data preparer.
UDFVersion Version;	The version of the UDF file system according to the UDFVersion (see page 205) enumeration.

UDFPartitionType PartitionType;	The partition type of the UDF file system according to the UDFPartitionType (see page 204) enumeration.
long MVDSAddress;	Main Volume Descriptor Sequence address.
long RVDSAddress;	Reserved Volume Descriptor Sequence address.
long RootAddress;	Root directory extent address.
long RootFEAddress;	Root directory File Entry address.
long PartitionAddress;	Partition address.
long PartitionLength;	Partition length.
long LVDAddress;	Logical Volume Descriptor address.
long PVDAddress;	Partition Volume Descriptor address.
long FSDAddress;	File System Descriptor address.
long VATAddress;	Virtual Allocation Table address.
long MetadataAddress;	Metadata partition address.
long SparingAddress;	Sparing area address.
long FileCount;	The number of files inside the UDF file system.
long DirCount;	The number of directories inside the UDF file system.
FileDateTime ^ RecordingDateTime;	The date / time info when the disc or images was created. You can overwrite it with FileDateTimeEx (see page 104).

Description

A structure that contains the information about the UDF information.

1.1.2.50 IsoSDK::WriteMethod Enumeration

C++

```
enum class WriteMethod {
  TrackAtOnce = 0,
  DiskAtOnce = 1,
  DiskAtOnce96 = 2
};
```

C#

```
public enum WriteMethod {
  TrackAtOnce = 0,
  DiskAtOnce = 1,
  DiskAtOnce96 = 2
}
```

File

Options.h

Members

Members	Description
TrackAtOnce = 0	Write method: TAO (Track-At-Once)
DiskAtOnce = 1	Write method: DAO (Disk-At-Once)
DiskAtOnce96 = 2	Write method: DAO96 (Disk-At-Once + 96)

Description

This enumeration defines the write methods the IsoSDK (see page 1) supports.

Index

A

B

C

D

E

F

G

H

I

J

L

M

T

U

V

W

Y